

# *Konzept*

# **Der heiße Draht**

*Ein dreitägiger Workshop  
im Deutschen Museum Bonn<sup>1</sup>  
19. bis 21. April 2017*

## **Ansprechpartner**

<b>Autor &amp; Workshop-Leiter</b>	Dr. Olav Schettler	<a href="mailto:olav@schettler.net">olav@schettler.net</a> <a href="https://github.com/tinkerthon/">github.com/tinkerthon/</a>
--	--------------------	---

## **Teilnehmer**

15 Schülerinnen und Schüler ab der 8. Klasse

## **Lizenz**



Das hier vorliegende Konzept ist unter der Lizenz "Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International"<sup>2</sup> verfügbar.

---

<sup>1</sup> Deutsches Museum Bonn, Ahrstraße 45, 53175 Bonn

<sup>2</sup> <http://creativecommons.org/licenses/by-sa/4.0/>

# Versionen

v1 - 2017-04-11	Erste vollständige Version
v2 - 2017-04-12	Web-Anwendung für Rangliste
v3 - 2017-04-15	Bereinigung
v4 - 2017-04-18	Liste der Vorbereitungen

# Inhalt

<b>Ansprechpartner</b>	<b>1</b>
<b>Teilnehmer</b>	<b>1</b>
<b>Lizenz</b>	<b>1</b>
<b>Versionen</b>	<b>2</b>
<b>Inhalt</b>	<b>2</b>
<b>Konventionen im Text</b>	<b>5</b>
<b>Informationen zum Workshop</b>	<b>5</b>
<b>Überblick</b>	<b>6</b>
<b>Vorbereitung</b>	<b>7</b>
<b>Detailplanung</b>	<b>7</b>
Stunde #1 - Begrüßung und Kennenlernen	7
Stunde #2 - Der BBC Microbit	8
Die Hardware auf der Platine kennenlernen	8
Eine Alternative aus Deutschland - der Calliope Mini	9
Erste Programmierübungen	10
Überblick über die Funktionen	13
Gruppe "Grundlagen"	14
Gruppe "Eingabe"	14
Gruppe "Musik"	14

Gruppe "LED"	15
Gruppe "Funk"	15
Gruppe "Schleifen"	15
Gruppe "Logik"	16
Gruppe "Platzhalter"	16
Gruppe "Mathematik"	16
Gruppe "Fortgeschritten"	16
Stunde #3 - Beispiele ausprobieren	16
Stunde #4 - "Der heiße Draht" im Block-Editor	16
Speichern der Berührungen	17
Extra Aufgabe	18
Berührungen zählen	18
Extra Aufgabe	20
Stunde #5 - Das Spiel bauen und ausprobieren	20
Stunde #6 - Den Micro:bit in Python programmieren	21
Warum Python?	21
Der Mu Editor	22
Stunde #7 - Einführung in Python	22
Die Read-Eval-Print-Loop REPL	22
Stunde #8 - Zugriff auf die Hardware des Micro:bit	27
Einfache Beispiele	27
Hello World	27
Bilder	27
Listen und Animationen	28
Taster	29
Ereignisschleifen	29
Ereignisse bedienen - Ein Cyber-Haustier	30
Ein- und Ausgänge - Kitzelig	30
Musik	31

Klangeffekte	32
Zufall	33
Zufallszahlen	33
Vorherbestimmte Zufallszahlen	34
Stunde #9 - Noch mehr Funktionen	34
Weitere Beispiele	34
Bewegung	34
Gesten	35
Magic-8-Ball	36
Richtung - Der Kompass	36
Dateisystem	37
Sprachausgabe	37
Netzwerk	37
Radio	37
Stunde #10 - Größere Programmbeispiele	37
Stunde #11 - Der heiße Draht in Python	39
Erste Version - Berührungen zählen	39
Zweite Version - Berührungen & Zeit zählen	39
Stunde #12 - Weitere Versionen	41
Dritte Version - Ergebnis anzeigen	41
Stunde #13 - Funkverbindung	42
Micro:bit als Glühwürmchen	42
Vierte Version: Ergebnisse an Zentrale für High Score melden	43
Fünfte Version: Eine zentrale Rangliste	44
<b>The End</b>	<b>48</b>
<b>Materialien</b>	<b>49</b>
<b>Historie des Workshops</b>	<b>50</b>
<b>Über den BBC micro:bit</b>	<b>51</b>
<b>Alternative Plattformen</b>	<b>52</b>
<b>Der heiße Draht - Physical Computing</b>	<b>52</b>

# Konventionen im Text

An einigen Stellen sind weiterführende Links als Fußnoten angegeben.

Eingerückte Absätze, die mit dem Bildschirm markiert sind, beschreiben Dinge hinter den Kulissen, die vorbereitet werden müssen.

 Besondere Hinweise sind mit dem Zeigefinger markiert.

Manchmal hat etwas nicht so funktioniert wie in der Anleitung oder Dokumentation beschrieben. Die Bombe weist darauf hin.

# Informationen zum Workshop

**Die Teilnehmer** lernen in diesem dreitägigen Workshop die Grundlagen der Programmierung von vernetzten Microcontrollern und damit das Internet der Dinge (IoT - Internet of Things) kennen. Der Bezug zur Erlebniswelt der Jugendlichen wird auf zwei Arten hergestellt. Zum einen zieht sich das bekannte Spiel „Der heiße Draht“ als wiederkehrendes Thema durch die gesamte Veranstaltung. Die Vorstellungsrunde schließt mit einem kleinen Turnier, in dem das Spiel mit seinem Aufbau und den Messgrößen Zeit und Berührungszählern kennengelernt werden.

Während der folgenden Einführung in Aufbau und Programmierung der Microcontroller BBC micro:bit kehren die Teilnehmer immer wieder zu diesem Spiel zurück und lernen schrittweise, wie Spielzeit und Berührungen gemessen, aufgezeichnet und schliesslich über ein drahtloses Netzwerk gesammelt und ausgewertet werden. Dieser Aufbau entspricht einen professionellen Meßföhernetzwerk und hat so direkten Bezug zur Berufswelt von Informationselektronikern.

Der zweite Bezug zur Erfahrungswelt der Teilnehmer liegt in den Komponenten des eingesetzten Microcontroller-Boards BBC micro:bit. Mit Accelerometer, Kompass, einen eingebauten Display und programmierbaren Tasten enthält das Board Sensoren und Aktoren, die Jugendliche vom alltäglichen Umgang mit ihren Smartphones als selbstverständlich erleben. In diesem Workshop lernen sie diese Komponenten auszulesen und über ein Netzwerk zu manipulieren.

**Der Dozent** ist promovierter Diplom-Informatiker mit über 25 Jahren Berufserfahrung. Der Workshop verknüpft Grundlagen der Programmierung und Netzwerktechnik mit dem praktischen Aufbau eines einfachen vernetzten Spiels. Auf diese Weise lernen die Teilnehmer die einzelnen Komponenten der eingesetzten Microcontroller-Boards aus eigener Erfahrung bei der Umsetzung einer konkreten Aufgabenstellung kennen. Die Programmierung der Microcontroller erfolgt in der professionellen Programmiersprache Python und vermittelt so einen realistischen Einblick in den Arbeitsalltag von Informatikern.

Die programmierbare Platine BBC micro:bit verbleibt nach dem Workshop bei den Teilnehmern. Diese haben so auch nach dem Workshop Gelegenheit, sich mit deren Anwendung zu beschäftigen und die Erweiterung mit externen Komponenten oder komplexere Aufgabenstellungen zu ergründen. Im Internet gibt es umfangreiche, didaktisch aufbereitete Materialien und Dokumentation zum BBC micro:bit.

**Der Workshop** wird im Rahmen des zdi-BSO-MINT-Programms gefördert. Er ist auf den MINT-(Ausbildungs-)Beruf *Informationselektroniker/in* und die Studiengänge *Internetbasierte Systeme* und *E-Services (grundständig)* ausgerichtet.

# Überblick

Der Workshop ist auf folgenden Seiten veröffentlicht:

- Website des Deutschen Museums Bonn<sup>3</sup>
- Blog-Beitrag auf Tinkerthon.de<sup>4</sup>
- Github-Repository<sup>5</sup>
- Facebook-Veranstaltung<sup>6</sup>
- Deutsches Python-Forum<sup>7</sup>

Der Workshop gliedert sich in 15 Einheiten zu je einer Stunde. Dazu gibt es drei Pausen. Wir machen nach je 50 Minuten 10 Minuten Pause.

<b>Mi 19.4.</b>	10	Stunde #1 - Begrüßung und Kennenlernen
	11	Stunde #2 - Der BBC Micro:bit
	12	Stunde #3 - Beispiele ausprobieren
	13	- Mittagspause in der Kantine -
	14	Stunde #4 - Der heiße Draht im Blockeditor
	15	Stunde #5 - Das Spiel bauen und ausprobieren
<b>Do 20.4.</b>	10	Stunde #6 - Den Micro:bit in Python programmieren
	11	Stunde #7 - Einführung in Python
	12	Stunde #8 - Zugriff auf die Hardware des Micro:bit
	13	- Mittagspause in der Kantine -

<sup>3</sup>

<http://www.deutsches-museum.de/bonn/information/fuer-kinder-und-schulen/die-kleine-eule-pfiffigus/workshops/der-heisse-draht/>

<sup>4</sup> <https://tinkerthon.de/workshops/2017/01/31/physical-computing-workshop-mit-bbc-microbit.html>

<sup>5</sup> <https://github.com/tinkerthon/Der-heisse-Draht-2017>

<sup>6</sup> <https://www.facebook.com/events/311698029247156/>

<sup>7</sup> <https://www.python-forum.de/viewtopic.php?f=21&t=40293>

	14	Stunde #9 - Noch mehr Funktionen
	15	Stunde #10 - Größere Programmbeispiele
<b>Fr 21.4.</b>	10	Stunde #11 - Der heiße Draht in Python
	11	Stunde #12 - Weitere Versionen
	12	Stunde #13 - Funkverbindung
	13	- Mittagspause in der Kantine -
	14	Stunde #14 - Freies Hacken
	15	Stunde #15 - Freies Hacken

# Vorbereitung

Auf jedem der Teilnehmer-Rechner

- Falls nötig: Installation eines aktuellen Chrome-Browsers<sup>8</sup>
- Installation des Mu-Editors<sup>9</sup>
- Installation des Uploader-Hilfsprogramms<sup>10</sup>
- Anlegen eines Ordners mit dem Inhalt des Github-Repos<sup>11</sup>

# Detailplanung

## Stunde #1 - Begrüßung und Kennenlernen

- [5m - Kursleiter]
  - Organisatorisches: Zeitplan, Mittagspausen, Handy nur in den Pausen (jede Stunde 10 Minuten)
- [10m] Der Kursleiter, die Helfer und die 15 Teilnehmer stellen sich reihum vor.
  - Name
  - Alter
  - Schule
  - Hat bereits Informatik oder Programmieren kennengelernt
  - Berufswunsch
  - Erwartung an diesen Workshop

<sup>8</sup> <https://www.google.com/chrome/browser/desktop/index.html>

<sup>9</sup> <https://codewith.mu/#download>

<sup>10</sup> <https://pxt.microbit.org/upload>

<sup>11</sup> <https://github.com/tinkerthon/Der-heisse-Draht-2017>

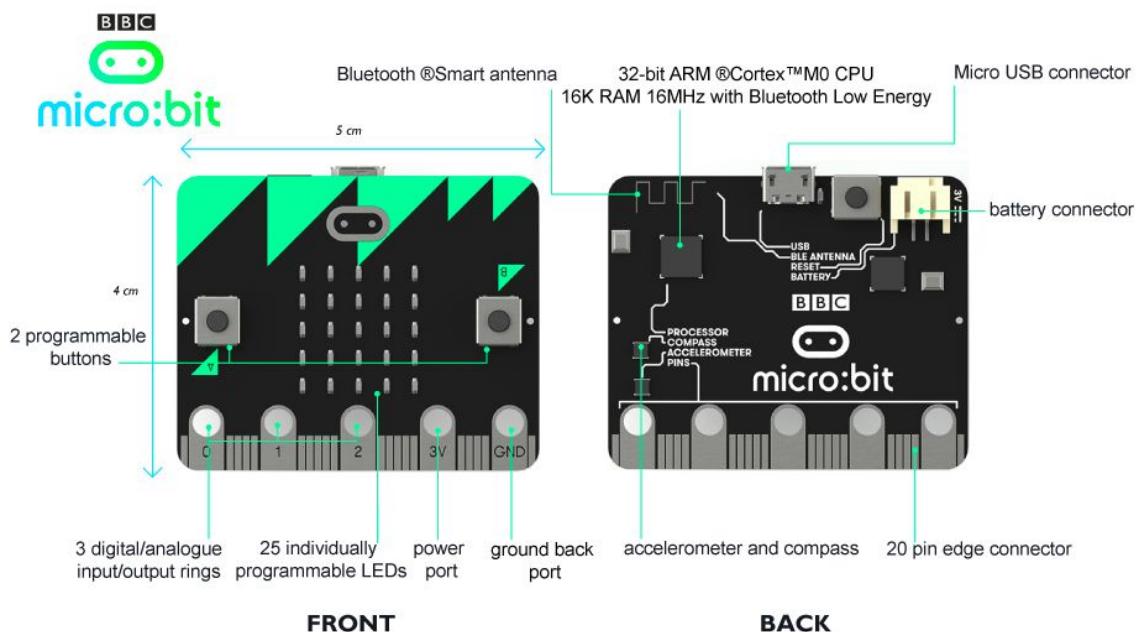
- [10m] Präsentation eines vorbereiteten Spielaufbaus
- Frage an die Teilnehmer: Wer kennt das Spiel “Der heiße Draht”? Wer kann die Spielregeln beschreiben?
- Mehrere Spieldurchläufe mit wechselnden Teilnehmern

## Stunde #2 - Der BBC Microbit

### Die Hardware auf der Platine kennenlernen<sup>12</sup>

Hier ein Überblick über die Hardware auf der Platine. Mehr Details einschließlich der Schaltpläne findest du auf der Website [tech.microbit.org](http://tech.microbit.org).<sup>13</sup>

- Das Gehirn: Eine 32-bit ARM Cortex M0 CPU mit 16kByte RAM und Bluetooth Low Energy (BLE)
- Sensoren: Accelerometer und Kompass, Temperatur und Helligkeit
- 25 rote LEDs
- Zwei programmierbare Taster A und B
- Ein weiterer Taster für Reset
- Eine gelbe Kontroll-LED auf der Rückseite
- Eine Micro-USB-Buchse
- Ein Batterieanschluss für ein Batterie-Pack mit zwei AAA-Batterien je 1,5V
- Drei digitale oder analoge Ein-/Ausgänge
- 20-poliger Kantenverbinder



<sup>12</sup> <http://microbit.org/de/hardware/>

<sup>13</sup> <http://tech.microbit.org/hardware/>

## Intro-Video und Bluetooth-Fähigkeiten

Hier ist ein Einführungsvideo<sup>14</sup> zur Platine:



Interessant sind die Möglichkeiten mit Bluetooth. Die Firma **Bitty Software** hat eine Android App<sup>15</sup>, Demo-Programme und viele Tutorials, um die Bluetooth-Fähigkeiten des Micro:bit auszuprobieren.<sup>16</sup>

## Eine Alternative aus Deutschland - der Calliope Mini

Im Herbst 2016 wurde eine erweiterte Variante des BBC Micro:bit aus Deutschland vorgestellt, der Calliope Mini.<sup>17</sup> Dieser kann Programme für den BBC Micro:bit verarbeiten und enthält im Kern die selben Bauteile wie der Micro:bit. Das sind die Unterschiede:

- Die Zielgruppe sind noch jüngere Kinder. Der BBC Micro:bit zielt auf 8-Klässler (ab 12 Jahren). Der Calliope Mini soll bereits an Drittklässler (8 Jahre) verteilt werden.
- Die Platine ist sternförmig; alle Bauteile liegen auf der Vorderseite. Dadurch kommt es bei Verwendung von Krokodilklemmen weniger zu Kurzschlüssen.
- Auf der Platine sind ein paar mehr Bauteile: Ein Lautsprecher / Microfon, eine bunte Leuchtdiode und zwei Modulstecker für Erweiterungen.

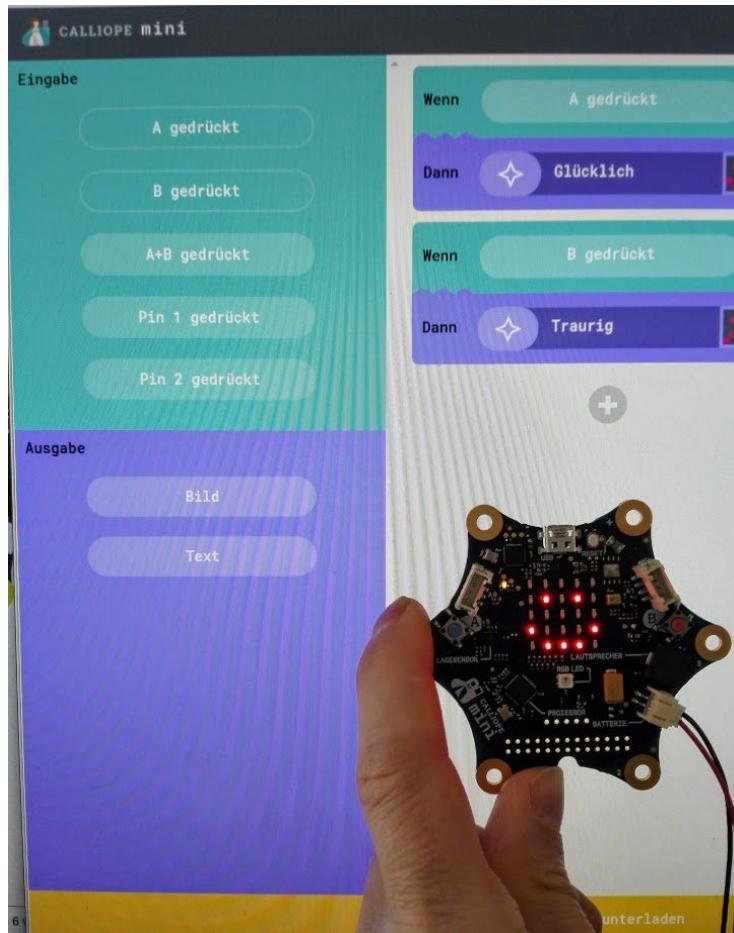
<sup>14</sup> Hier ein kurzes Video zu der Platine: <https://www.youtube.com/watch?v=Wuza5WXiMkc>

<sup>15</sup> <https://play.google.com/store/apps/details?id=com.bluetooth.mwoolley.microbitledemo>

<sup>16</sup> <http://www.bittyssoftware.com/downloads.html>

<sup>17</sup> <http://calliope.cc/ueber-mini>

- Neben den Programmierumgebungen für den BBC Micro:bit gibt es zwei weitere:<sup>18</sup>  
Eine stark vereinfachte Umgebung für jüngere Kinder und eine, die auf dem Editor NEPO von Roberta / Fraunhofer basiert.
- Die Dokumentation<sup>19</sup> und einen angepassten PXT-Editor gibt es ansatzweise in Deutsch.



## Erste Programmierübungen

Grundsätzlich ist der Micro:bit über eine Weboberfläche programmierbar.<sup>20</sup> Es gibt zwei Editoren auf der Website. Wir starten für erste Experimente mit dem Blocks Editor (PXT).



Bei den folgenden Experimenten muss sehr oft eine HEX-Datei auf den Micro:bit kopiert werden. Das wird schnell lästig. Du kannst aber ein kleines Hilfsprogramm<sup>21</sup> installieren, das dir diese Arbeit abnimmt.

<sup>18</sup> <https://calliope.cc/editor>

<sup>19</sup> <http://pxt.calliope.cc/reference>

<sup>20</sup> <https://microbit.org/de/code/>

<sup>21</sup> <https://pxt.microbit.org/uploader>

Um unabhängig vom WLAN zu sein, benutzen wir für die ersten Schritte eine lokale Installation des Blockeditors MakeCode<sup>22</sup> von Microsoft. Die erforderliche Server-Komponente ist OpenSource und läuft auf dem Linux-Rechner des Workshop-Leiters. Hier sind die Schritte für eine Installation<sup>23</sup>:

```
# Installiere Node.JS
sudo apt-get install npm
sudo npm install -g n
sudo n latest

# Installiere Abhängigkeiten von PXT
sudo apt-get install libgnome-keyring-dev

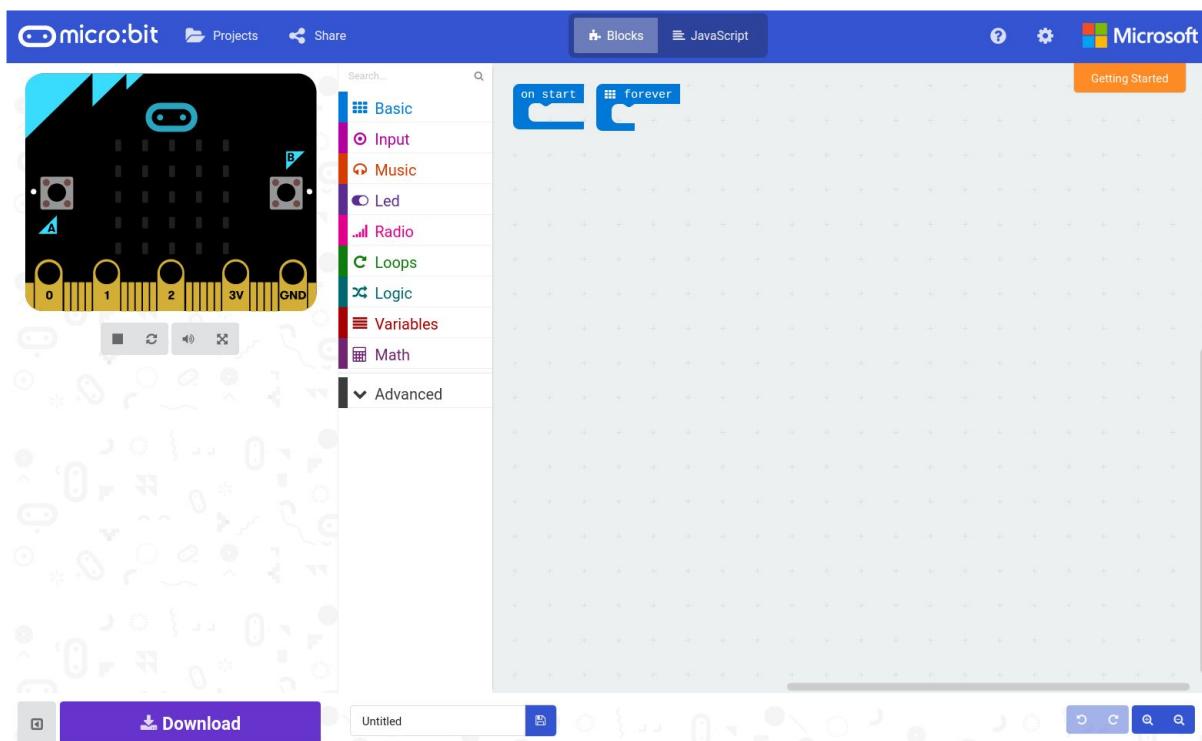
# Installiere PXT
sudo npm install -g pxt
cd ~/Documents/osterworkshop-2017/
mkdir microbit
cd microbit
pxt target microbit

# Starte die Umgebung
pxt serve
```

---

<sup>22</sup> <https://makecode.com/>

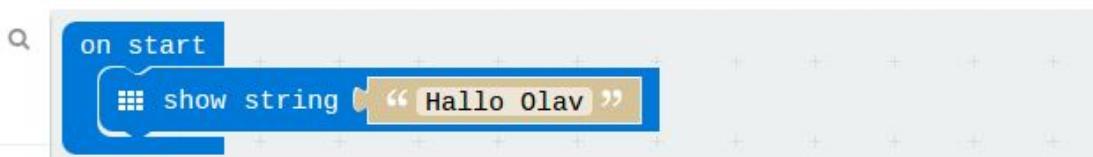
<sup>23</sup> Installation unter Elementary OS 0.4 Loki (64 Bit), basierend auf Ubuntu 16.04.2 LTS



Damit diese Umgebung auch von den Rechnern der Teilnehmer aus zu erreichen ist, muss auf dem Rechner des Workshop-Leiters ein Proxy laufen. Dazu eignet sich z.B. NGINX.

Die Umgebung enthält ein Tutorial, welches die Teilnehmer jeweils zu zweit durcharbeiten können.

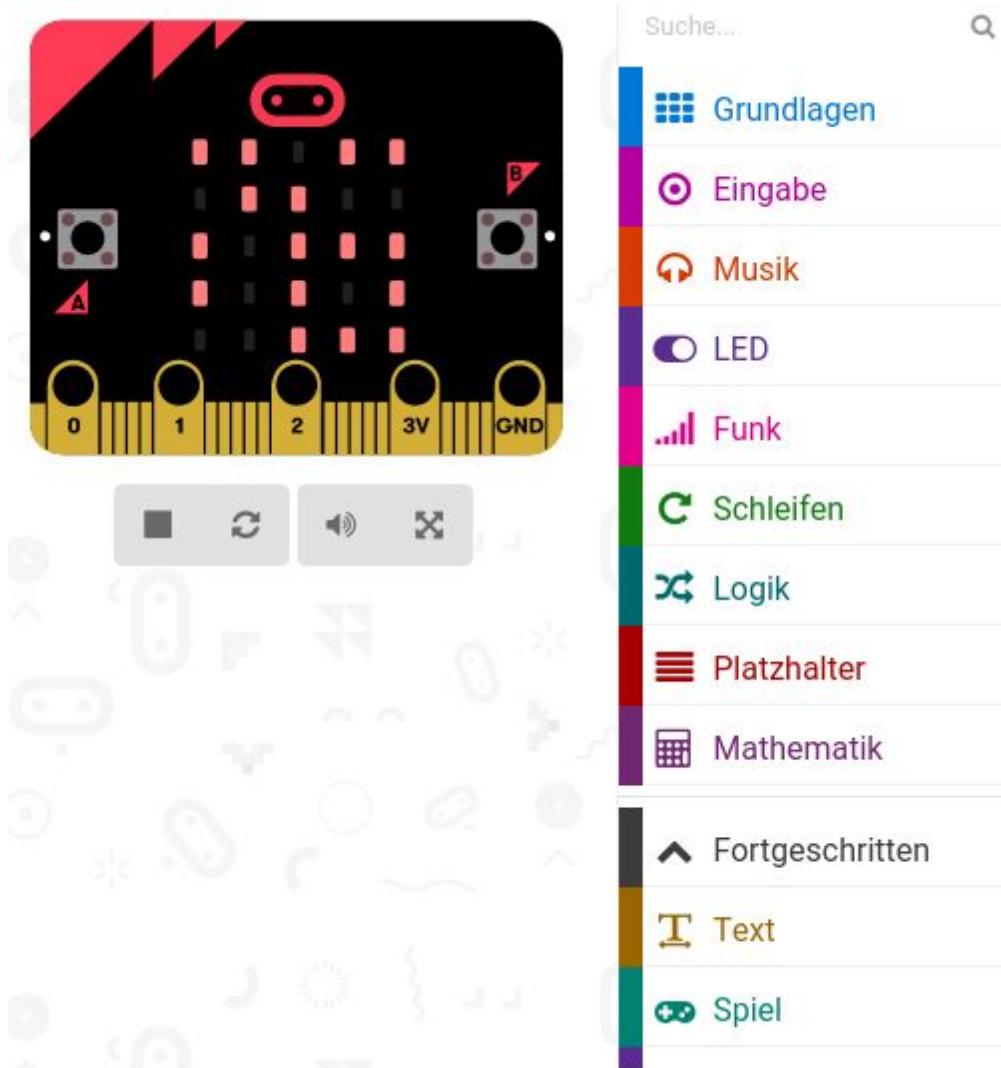
Welcome! Place the **show string** block in the **on start** slot to scroll your name.



Die kleinen Übungen zeigen Laufschrift und erklären, wie man auf Drücke der Taster reagiert. Das letzte Beispiel realisiert einen Würfel, der beim Schütteln eine zufällige Zahl zwischen 1 .. 6 zeigt:



## Überblick über die Funktionen



Die Programmieroberfläche verteilt die einzelnen Funktionen auf 9 Gruppen und ein Aufklappmenü "Fortgeschritten". Dadurch ist es manchmal etwas schwierig, eine gesuchte Funktion zu finden. Wie schauen im folgenden kurz in jede Gruppe hinein, um einen Überblick über deren Inhalt zu bekommen.



Alle Funktionen sind ausführlich in der Dokumentation<sup>24</sup> erklärt. Die Dokumentation ist allerdings noch weitgehend auf Englisch. Du kannst aber selber mithelfen, die Dokumentation zu übersetzen.<sup>25</sup>

<sup>24</sup> <https://pxt.microbit.org/docs>

<sup>25</sup> <https://makecode.com/translate>

## **Gruppe “Grundlagen”**

Module für die allerersten Schritte. Funktionen zum Anzeigen von Bildfolgen und Laufschrift auf den 25 LEDs (dem “Bildschirm” des Micro:bit).

## **Gruppe “Eingabe”**

Reagieren auf Eingaben und die Umgebung:

- Taster
- Beschleunigungssensor
- Kompass
- Helligkeit
- Temperatur
- Eingangssignale an den Pins
- Laufzeit

## **Gruppe “Musik”**

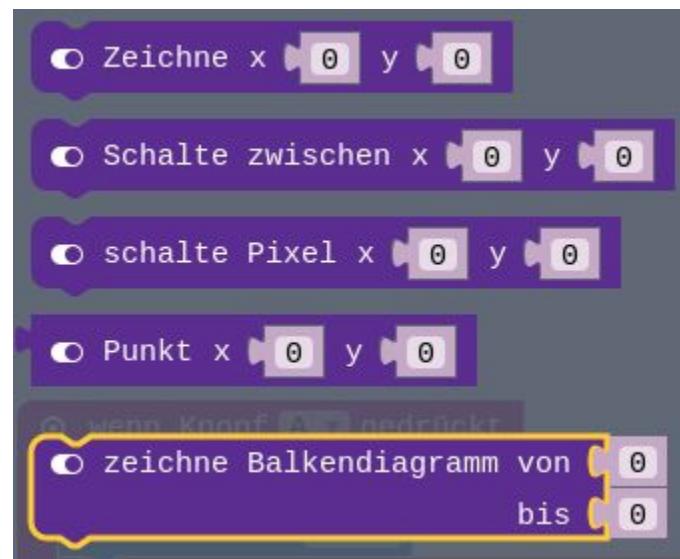
Funktionen zum Erzeugen von Tönen und Melodien:

- Eingebaute Melodien und Tonfolgen
- Tonhöhe
- Taktlänge
- Geschwindigkeit

Da der Micro:bit keinen eingebauten Lautsprecher hat, schließt man für diese Funktionen zwischen den Pins 1 und GND einen Lautsprecher an. Der Calliope Mini hat einen Lautsprecher eingebaut. Hier funktionieren die Musikfunktionen ohne weitere Hardware.

Die Funktion *Melodie* hat bei mir nicht funktioniert.

## Gruppe “LED”



Funktionen für die 5x5 Matrix aus roten Leuchtdioden (LED - “Light emitting diode”), dem “Bildschirm” des BBC Micro:bit. Die deutsche Übersetzung ist etwas eigenwillig; hier die Erklärungen aus der Dokumentation:<sup>26</sup>

- Zeichne (“plot”). Schalte einzelne LED **ein** (x - horizontal, y - vertikal)
- Schalte zwischen (“toggle”). Schalte einzelne LED **um**
- Schalte Pixel (“unplot”). Schalte einzelne LED **aus**
- Punkt. Ermittle den Zustand einer einzelnen LED (ein oder aus)
- Zeichne Balkendiagramm. Der erste Wert gibt die Höhe bezogen auf den zweiten Wert an

## Gruppe “Funk”

Funktionen, um per Funk zwischen mehreren Micro:bit Daten auszutauschen. Das verwendete Verfahren ist nicht mit Bluetooth kompatibel und funktioniert nur zwischen Micro:bits und nicht z.B. zwischen Micro:bit und Handy:

- Senden und Empfangen von Zahlen und Texten
- Teilnehmen an einer Funkgruppe

## Gruppe “Schleifen”

Funktionen zur Ablaufsteuerung:

- Eine bestimmte Anzahl mal wiederholen
- Wiederholen, solange eine Bedingung erfüllt ist

<sup>26</sup> <https://pxt.microbit.org/reference/led>

## **Gruppe “Logik”**

Funktionen, um auf Bedingungen zu reagieren:

- Wenn .. dann
- Vergleich von Werten
- “Und” und “oder”
- Wahrheitswerte “Wahr” und “Falsch”
- 

## **Gruppe “Platzhalter”**

Auch “Variable” genannt, sind diese ein wichtiges Konzept in vielen Programmiersprachen. Eine Variable ist ein benannter Ablageort für Werte. Über die Funktionen in dieser Gruppe kann

- der Wert an einem Ablageort ermittelt werden,
- ein neuer Wert gesetzt oder
- ein Wert erhöht (oder erniedrigt) werden

## **Gruppe “Mathematik”**

Funktionen für Grundrechenarten, Rest, einfache Zahlenwerte, Zufallszahlen, Minimal-Maximal- und Absolutwerte. Auch eine Funktion zur Umwandlung von Zahlen in Zeichen (eine Dekodierung)

## **Gruppe “Fortgeschritten”**

Zusätzlich können auch noch weitere Bibliotheken mit Erweiterungen nachgeladen werden. Damit lassen sich z.B. angeschlossene Geräte wie Adafruit NeoPixel oder auch das eingebaute Bluetooth nutzen.

## **Stunde #3 - Beispiele ausprobieren**

In der Dokumentation<sup>27</sup> sind viele Beispiele enthalten. Schau sie dir an, probiere einige davon im Simulator oder lade sie auf deinen Micro:bit.

## **Stunde #4 - “Der heiße Draht” im Block-Editor**

In dieser Stunde kehren wir wieder zu unserem Spiel zurück und realisieren eine Steuerung mit den Elementen des Block-Editors, die wir in den vergangenen Stunden kennengelernt haben. Wir folgen dabei lose einer Anleitung der RaspberryPi Foundation.<sup>28</sup>

---

<sup>27</sup> <https://pxt.microbit.org/docs>

<sup>28</sup> <https://codeclubprojects.org/en-GB/microbit/frustration/>

Hier sind die Schritte:

## Speichern der Berührungen

Starte ein neues Projekt und nenne es "heisser-draht"

Entferne die beiden Blöcke "beim Start" und "dauerhaft"



Mit Drücken des Tasters A soll ein neues Spiel beginnen. Ziehe "Wenn Knopf A gedrückt" aus der Gruppe "Eingabe".

Nun brauche wir eine Variable, um die Anzahl von Berührungen festzuhalten. Klicke auf "Platzhalter" und dann auf "Neuen Platzhalter anlegen". Gib dem Platzhalter den Namen "berührt". Ziehe dann den Block "ändere ... auf" in den Block "Wenn Knopf ..." und wähle "berührt" aus:

Damit wird die Zahl der gezählten Berührungen beim Drücken der Taste A auf 0 gesetzt. Die Zahl der Berührungen wollen wir anzeigen. Ziehe dazu den Block "Zeige Nummer" aus "Grundlagen" und setze darin "berührt" aus "Platzhalter" ein:

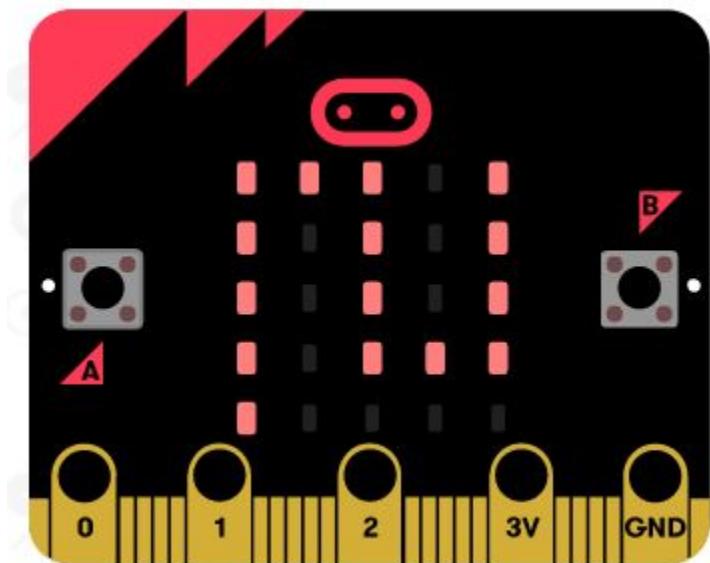
Nun kannst du das Programm schon im Simulator ausprobieren:

Anklicken von Knopf A sollte als Anzahl Berührungen 0 anzeigen.



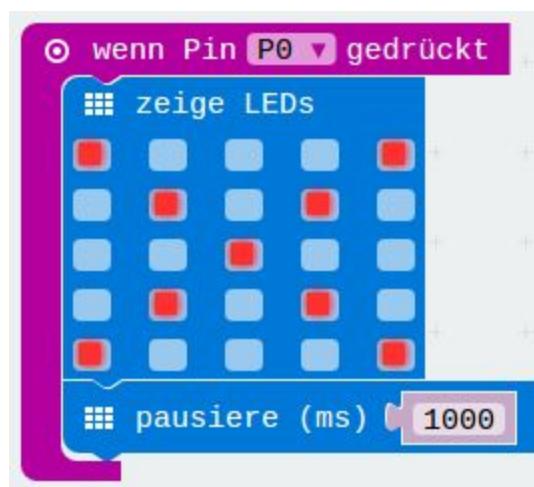
## Extra Aufgabe

Kannst du für eine Sekunde ein Bild anzeigen, bevor die Zahl der Berührungen angezeigt wird:



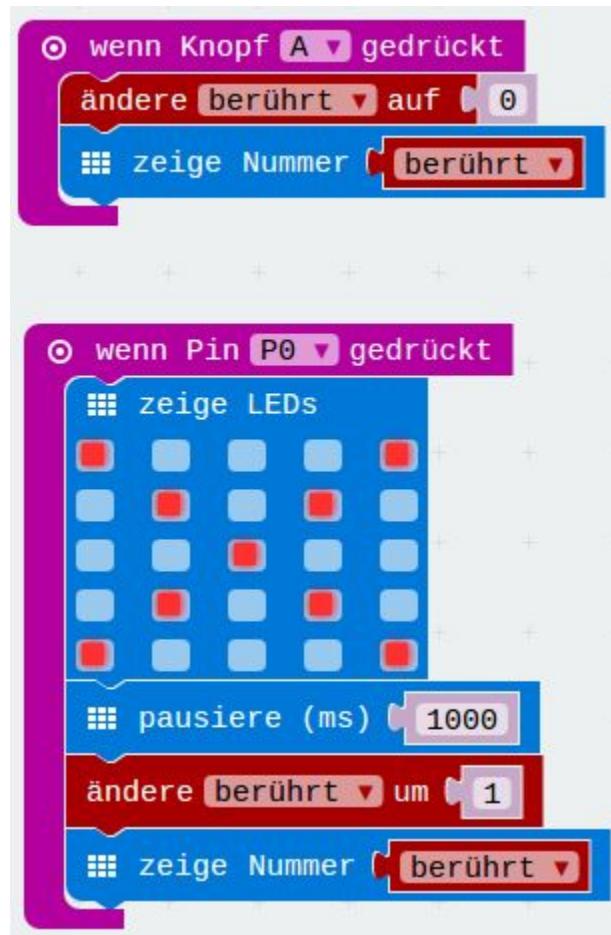
## Berührungen zählen

Als nächstes werden wir bei jeder Berührung an Pin 0 die Variable "berührt" um 1 erhöhen. Ziehe hierzu "Wenn Pin0 gedrückt" aus "Eingabe". Füge als nächstes zwei Blöcke zum Anzeigen eines Kreuzes für 1 Sekunde hinzu. Das Ergebnis sieht so aus:

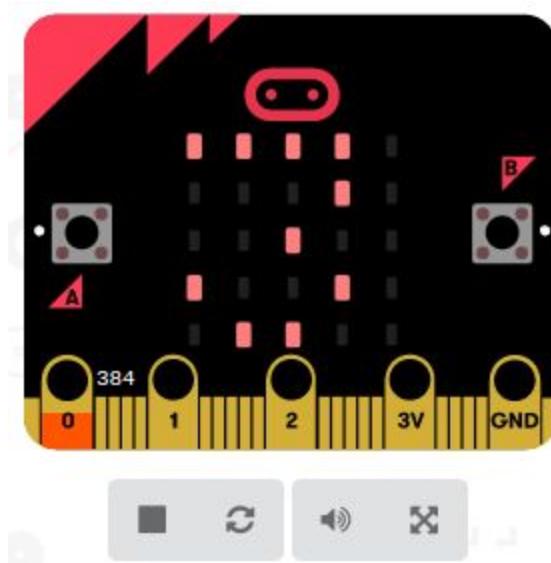


1 Sekunde sind 1000 Millisekunden. Die roten Punkte des Kreuzes setzt du durch Draufklicken.

Jetzt brauchen wir einen Block zum Zählen der Berührungen und einen weiteren zur Anzeige. Der vollständige Code unserer Spielsteuerung sieht jetzt so aus:



Nun können wir wieder testen. Drücke A im Simulator, um das Spiel zu starten. Jeder Druck auf Pin 0 zeigt nun kurz ein "X" und dann die Anzahl der Berührungen:



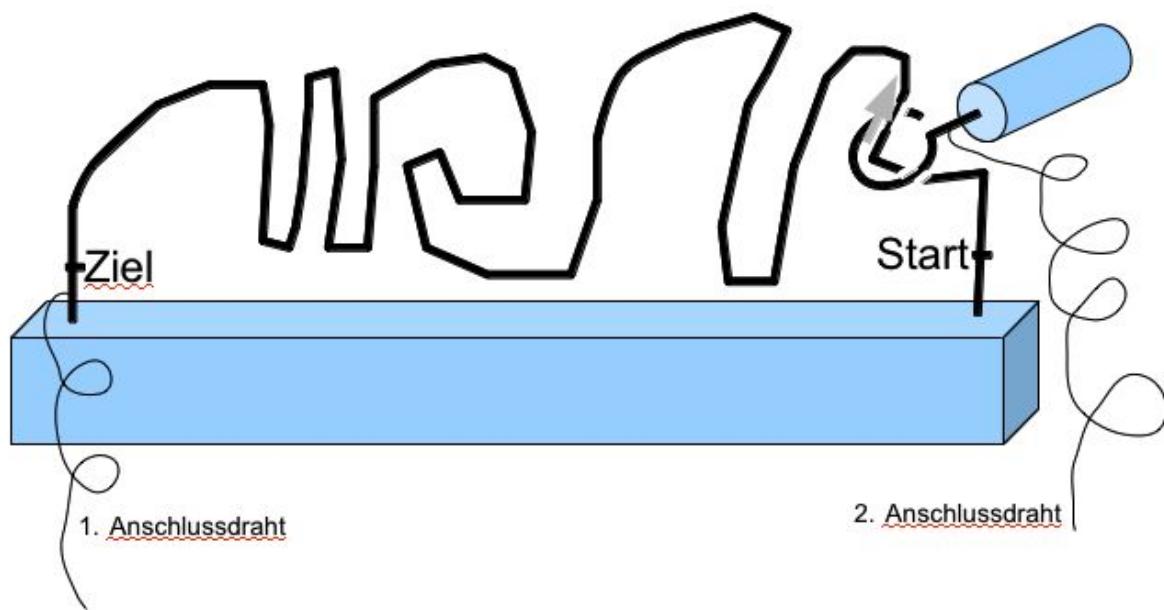
Klicke nun auf "Herunterladen" und kopiere das Skript auf deinen Micro:bit. Du kannst die Verbindung zwischen Pin 0 und GND über zwei Finger auf diesen Pins herstellen.

## Extra Aufgabe

Kannst du einen Cheat einbauen, bei dem ein Druck auf Taster B die Berührungen um 1 reduziert?

## Stunde #5 - Das Spiel bauen und ausprobieren

Jetzt bauen wir das Spiel auf und probieren damit unsere Spielsteuerung. Die folgende Skizze zeigt das Ergebnis:

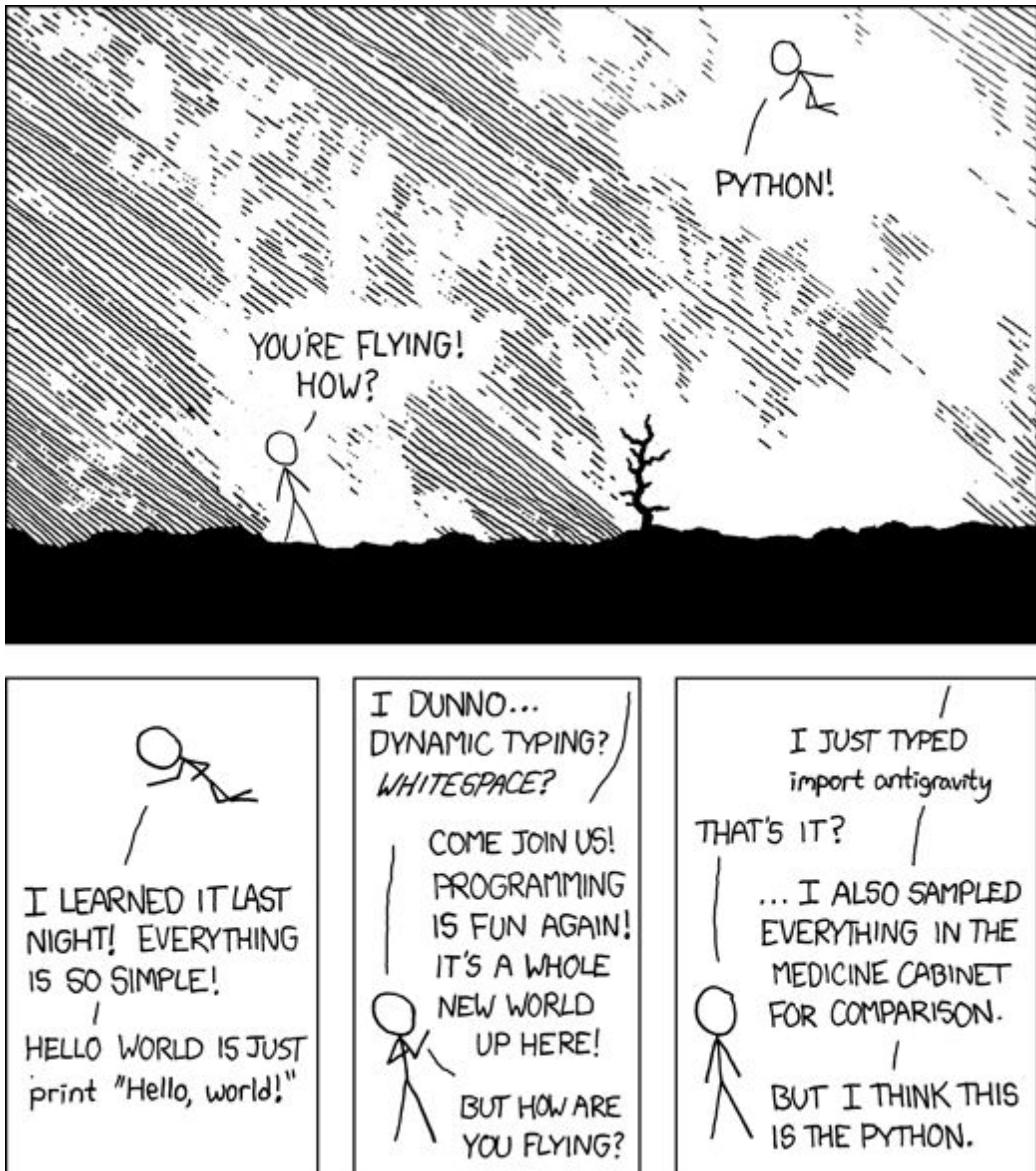


Hier sind die Schritte:

- Schneide ein 30 cm langes Stück Draht und stecke es in die beiden Bohrungen im großen Holzstück. Das ist der Spieldraht. Du kannst einige Buchten oder Schleifen in den Draht biegen, um das Spiel schwieriger zu machen. Aber übertreibe es nicht.
- Als nächstes schneide ein 10 cm langes Drahtstück und biege es zu einer Öse mit einer Öffnung von 1 cm. Stecke diese in die Bohrung in dem Rundstab.
- Fixiere alle Drahtstücke mit einem Tropfen Leim in alle Bohrungen.
- Verbinde nun ein erstes Kabel mit zwei Krokodilklemmen am einen Ende des Spieldrahtes und mit dem anderen Ende mit dem GND-Pin an deinem Micro:bit.
- Verbinde zur Verlängerung zwei Kabel über Krokodilklemmen mit einander. Verbinde das eine Ende mit der Öse, das andere Ende mit dem Pin 0 deines Micro:bit.

Das war's! Jetzt könnt ihr ein "Heißer Draht"-Turnier veranstalten. Du kannst dein Spiel tunen, indem du einen längeren Spieldraht einsetzt, mehr Kurven oder sogar Loopings einbaust oder die Schlaufe enger biegst.

## Stunde #6 - Den Micro:bit in Python programmieren



### Warum Python?

Python ist eine weit verbreitete Programmiersprache. Das Comic<sup>29</sup> zeigt, was sie kann ;)  
Comics selber machen?<sup>30</sup>

Mit Python kann man nicht nur den Micro:bit programmieren sondern auch:

- Web-Anwendungen mit Django<sup>31</sup> oder Flask<sup>32</sup>
- Wissenschaftliches Arbeiten mit Pandas<sup>33</sup> oder IPython<sup>34</sup>

<sup>29</sup> <https://xkcd.com/353/>

<sup>30</sup> <http://pycomic.github.io/>

<sup>31</sup> <http://www.djangoproject.com/>

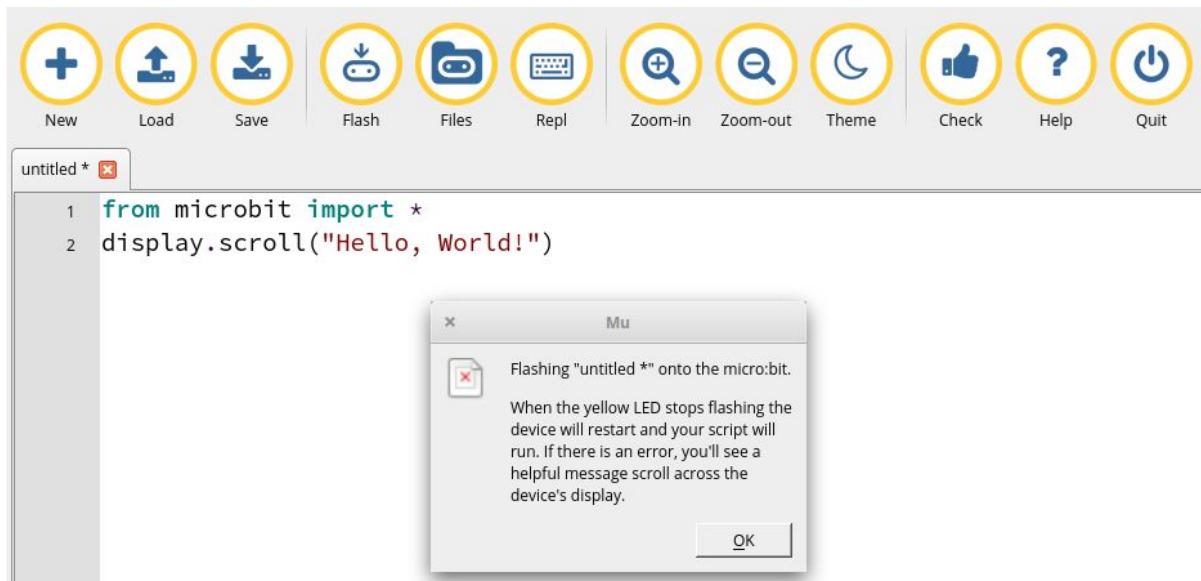
<sup>32</sup> <http://flask.pocoo.org/>

<sup>33</sup> <http://pandas.pydata.org/>

- Instagram<sup>35</sup>, Pinterest<sup>36</sup>, YouTube<sup>37</sup>, Google und Dropbox<sup>38</sup>

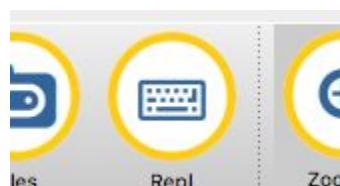
## Der Mu Editor

Der Editor läuft direkt auf deinem Rechner. Mit angeschlossenem Micro:bit kannst du das übersetzte Programm direkt auf den Micro:bit übertragen.



## Stunde #7 - Einführung in Python

### Die Read-Eval-Print-Loop REPL



Der Mu-Editor gibt dir einen direkten Zugang zum Python-Interpreter (=Übersetzer) auf dem Micro:bit:

<sup>34</sup> <http://ipython.org/>

<sup>35</sup> <https://www.quora.com/What-programming-languages-are-used-at-Instagram>

<sup>36</sup> <https://www.quora.com/What-is-the-technology-stack-behind-Pinterest-1>

<sup>37</sup> <https://www.quora.com/Is-YouTube-using-Python-to-develop>

<sup>38</sup> [https://de.wikipedia.org/wiki/Guido\\_van\\_Rossum](https://de.wikipedia.org/wiki/Guido_van_Rossum)

```
Traceback (most recent call last):
  File "<__main__>", line 9, in <module>
KeyboardInterrupt:
MicroPython v1.7-9-gbe020eb on 2016-04-18; micro:bit with nRF51822
Type "help()" for more information.
>>> help()
Welcome to MicroPython on the micro:bit!

Try these commands:
display.scroll('Hello')
running_time()
sleep(1000)
button_a.is_pressed()
```

Es folgt ein Python-Schnellkurs. Willst du Python ohne einen Micro:bit ausprobieren, geht das am einfachsten über eine Online-Umgebung, z.B. [pythonfiddle](http://pythonfiddle.com)<sup>39</sup> oder [pythonanywhere](https://www.pythonanywhere.com/)<sup>40</sup>



Du kannst Python in der REPL wie einen Taschenrechner benutzen:

```
>>> 2 * 3.14 * 15
94.19998
```

Nicht nur mit Zahlen kannst du rechnen, sondern auch mit Texten:

```
>>> "Hallo" + "Welt"
"HelloWelt"
```

Du kannst Dinge vergleichen:

```
>>> 42 > 100
False
```

<sup>39</sup> <http://pythonfiddle.com/>

<sup>40</sup> <https://www.pythonanywhere.com/>

```
>>> "Abc" < "Bcd"
```

```
True
```

```
>>> "eins" == 1
```

```
False
```

Manchmal macht ein Vergleich keinen Sinn. Dann meckert MicroPython:

```
>>> 42 > "hundert"
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: unsupported types for __gt__: 'int', 'str'
```

Neben Zahlen und Texten ("Strings") gibt es Listen:

```
>>> [1, "2", "drei"]
```

```
[1, '2', 'drei']
```

Ein paar Easter Eggs (verborgene Funktionen) sind auch in MicroPython versteckt:

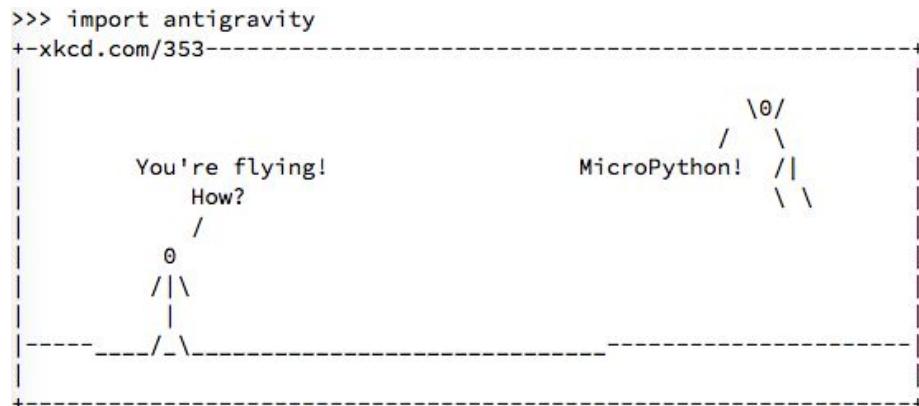
```
>>> help()
```

```
... Available module: antigravity, array, collections, gc, love,  
math,  
micropython, music, neopixel, os, radio, random, speech, struct,  
sys, this
```

```
>>> import love
```

Ein Herz blinkt auf dem Bildschirm :)

Oder dieses hier:



Mit `dir()` lassen sich Dinge auflisten:

```
>>> dir(love)
```

```
['__name__', '__init__', 'badaboom']
```

```
>>> love.badaboom()
```

Auch Texte enthalten Dinge:

```
>>> dir("hallo Welt!")
['find', 'rfind', 'index', 'rindex', 'join', 'split', 'rsplit',
'startswith', 'endswith', 'strip', 'lstrip', 'rstrip', 'format',
'replace', 'count', 'partition', 'rpartition', 'lower', 'upper',
'isspace', 'isalpha', 'isdigit', 'isupper', 'islower']
```

Diese Dinge sind Methoden, also Verhalten, die sich auf den Text auswirken:

```
>>> "Hallo Welt!".count("l")
3
>>> "Hallo Welt!".upper()
"HALLO WELT!"
```

Hier sind die Methoden von Listen:

```
>>> dir([1, "2", "drei"])
['append', 'clear', 'copy', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']

>>> liste = [1, 2, 3]
>>> liste.reverse()
>>> liste
[3, 2, 1]

>>> liste.pop()
1

>>> liste.append("fortytwo")
>>> liste
[3, 2, 'fortytwo']
```

Manchmal passieren Fehler:

```
>>> zeit
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name zeit is not defined
```

Du kannst neue **Variable** definieren und ihren Wert abfragen:

```
>>> running_time()
3014964

>>> zeit = running_time()
>>> zeit
3017160
```

```
>>> zeit = zeit + 1  
3017161
```

```
>>> zeit += 1  
3017162
```

Du kannst Dinge mehrmals tun:

```
>>> for zahl in range(1, 8, 2):  
...     Zahl  
...  
1  
3  
5  
7
```

... oder solange, bis ein Ergebnis eintritt:

```
>>> zahl  
12  
>>> while zahl > 7:  
...     zahl = zahl - 1  
...     zahl  
...  
11  
10  
9  
8
```

Willst du eine Liste von Werten, so kannst du Listen und for-Schleifen kombinieren:

```
>>> [i*2 for i in range(8)]  
[0, 2, 4, 6, 8, 10, 12, 14]
```

Du kannst dem Micro:bit Dinge beibringen. Wir wollen eine Vorschrift zur Berechnung der **Fakultät** einer Zahl aufschreiben. Solche Vorschriften heissen im Python selbstdefinierte **Funktionen**. Ein paar eingebaute Funktionen haben wir oben schon kennengelernt: `help()`, `dir()`. Hier ist eine Funktion, um die Fakultät einer Zahl<sup>41</sup> zu berechnen. Die Fakultät einer Zahl  $n!$  gibt Anzahl der Möglichkeiten an,  $n$  unterscheidbare Gegenstände in einer Reihe anzuordnen.

$$4! = 1 * 2 * 3 * 4$$

Hier das Beispiel auf dem Micro:bit:

```
>>> def fakultaet(n):  
...     f = 1  
...     for i in range(1, n + 1):
```

---

<sup>41</sup> [https://de.wikipedia.org/wiki/Fakult%C3%A4t\\_\(Mathematik\)#Pythonprogramm](https://de.wikipedia.org/wiki/Fakult%C3%A4t_(Mathematik)#Pythonprogramm)

```
...     f = f * i
...     return f
...
>>> fakultaet(4)
24
```

## Stunde #8 - Zugriff auf die Hardware des Micro:bit

### Einfache Beispiele

Die komplette MicroPython-Dokumentation findest du im Web.<sup>42</sup> Dort kannst du dir auch eine EPUB-Datei für deinen eBook-Reader oder eine PDF-Datei zum Ausdrucken<sup>43</sup> besorgen. Willst du mehr über MicroPython erfahren, so gibt es auf [tech.microbit.org](http://tech.microbit.org)<sup>44</sup> eine eigene Seite dazu.

MicroPython ist eine spezielle Variante von Python für Microcontroller. Neben der englischsprachigen Originaldokumentation<sup>45</sup> gibt es auch ein deutsches Tutorial.<sup>46</sup>

Wir fangen mit einfachen Beispielen an.

### Hello World

```
from microbit import *
display.scroll("Hello, World")
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/hello.html#hello-world>

- Ein Modul importieren
- Eine Methode mit Argumenten aufrufen

### Bilder

```
from microbit import *
display.show(Image.HAPPY)
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html>

- Methode *show* aus dem *Image*-Objekt
- Eingebaute Bilder

<sup>42</sup> <https://microbit-micropython.readthedocs.io/en/latest/>

<sup>43</sup> <https://media.readthedocs.org/pdf/microbit-micropython/latest/microbit-micropython.pdf>

<sup>44</sup> <http://tech.microbit.org/software/micropython/>

<sup>45</sup> <https://docs.python.org/3.5/>

<sup>46</sup> <https://py-tutorial-de.readthedocs.io/de/python-3.3/index.html>

```
from microbit import *

boat = Image("05050:"
            "05050:"
            "05050:"
            "99999:"
            "09990")

display.show(boat)
```

- Selbst erstellte Bilder

## Listen und Animationen

```
from microbit import *
display.show(Image.ALL_CLOCKS, loop=True, delay=100)
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html#animation>

- Listen in Python: `mixed_up_list = ["hello!", 1.234, Image.HAPPY]`

```
from microbit import *

boat1 = Image("05050:"
              "05050:"
              "05050:"
              "99999:"
              "09990")

boat2 = Image("00000:"
              "05050:"
              "05050:"
              "05050:"
              "99999")

boat3 = Image("00000:"
              "00000:"
              "05050:"
              "05050:"
              "05050")
```

```

boat4 = Image("00000:"
              "00000:"
              "00000:"
              "05050:"
              "05050")

boat5 = Image("00000:"
              "00000:"
              "00000:"
              "00000:"
              "05050")

boat6 = Image("00000:"
              "00000:"
              "00000:"
              "00000:"
              "00000")

all_boats = [boat1, boat2, boat3, boat4, boat5, boat6]
display.show(all_boats, delay=200)

```

- Selbst erstellte Animation
- Weitere Ideen: Eigene Animationen. Special Effects. Fade out / in

## Taster

```

from microbit import *

sleep(10000)
display.scroll(str(button_a.get_presses()))

```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/buttons.html#buttons>

- Pausen,
- Objekt button\_a
- Umwandlung von Zahl in Text mit str()
- Geschachtelte Argumente

## Ereignisschleifen

```

from microbit import *

while running_time() < 10000:
    display.show(Image.ASLEEP)

```

```
display.show(Image.SURPRISED)
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/buttons.html#event-loops>

- Funktion running\_time()
- Geschachtelter Code

## Ereignisse bedienen - Ein Cyber-Haustier

```
from microbit import *

while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    elif button_b.is_pressed():
        break
    else:
        display.show(Image.SAD)

display.clear()
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/buttons.html#handling-an-event>

- Endlosschleife
- If .. elif .. else
- button\_b.is\_pressed()

## Ein- und Ausgänge - Kitzelig

```
from microbit import *

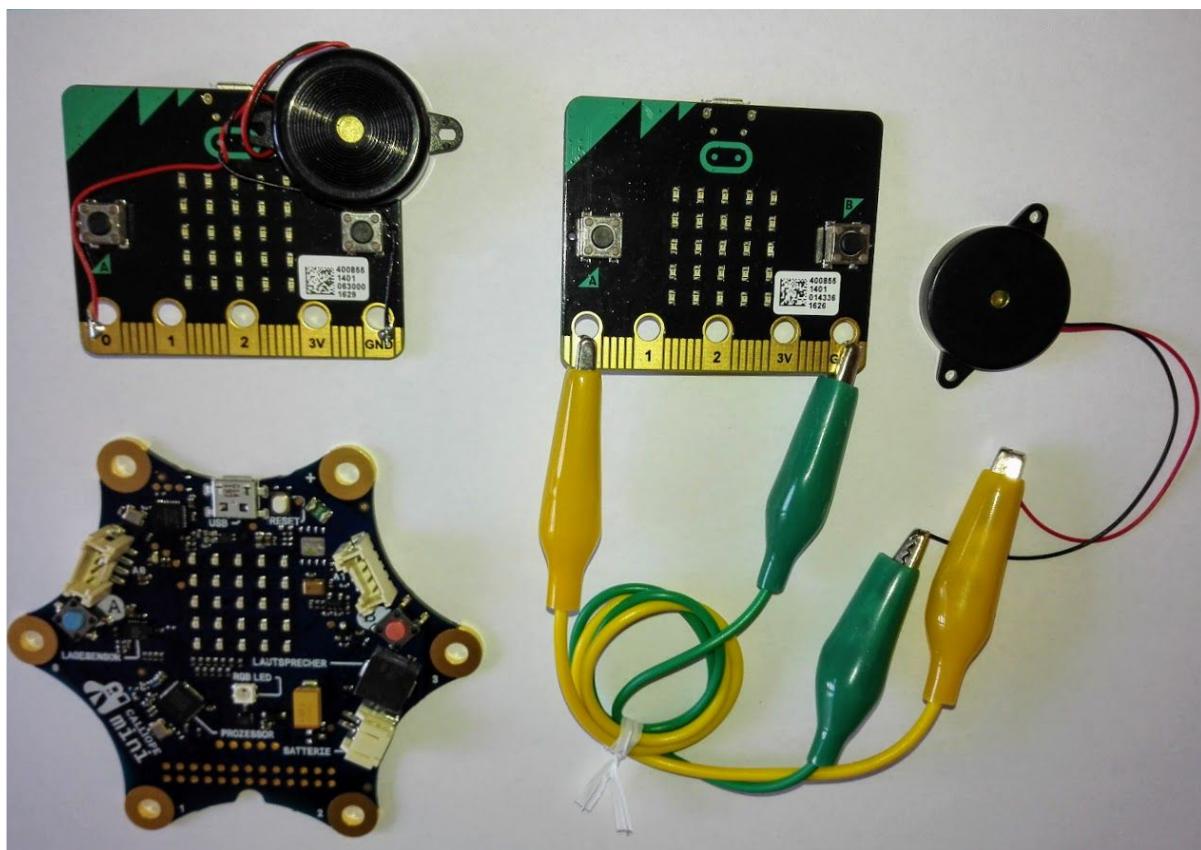
while True:
    if pin0.is_touched():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/io.html#ticklish-python>

- pin0.is\_touched()

## Musik

Damit der folgende Code funktioniert, mußt du einen Lautsprecher zwischen die Pins 0 und GND anschließen:



Oben rechts im Bild habe ich einen Piezo-Lautsprecher mit Krokodilklemmen zwischen den Pins 0 und GND angeschlossen. Oben links habe ich den Lautsprecher angelötet und ihn mit einem kleinen Stück Velcro auf der Platine befestigt. Der Calliope Mini unten links hat bereits einen Lautsprecher eingebaut.

Micropython hat einige eingebaute Melodien, z.B. Nyan.cat<sup>47</sup>

```
import music
music.play(music.NYAN)
```



<https://microbit-micropython.readthedocs.io/en/latest/tutorials/music.html#music>

<sup>47</sup> <http://www.nyan.cat/>

Hier ist die vollständige Liste eingebauter Musikstücke:

- music.DADADADUM
- music.ENTERTAINER
- music.PRELUDE
- music.ODE
- music.NYAN
- music.RINGTONE
- music.FUNK
- music.BLUES
- music.BIRTHDAY
- music.WEDDING
- music.FUNERAL
- music.PUNCHLINE
- music.PYTHON
- music.BADDY
- music.CHASE
- music.BA\_DING
- music.WAWAWAWAA
- music.JUMP\_UP
- music.JUMP\_DOWN
- music.POWER\_UP
- music.POWER\_DOWN

Du kannst auch selber Musikstücke komponieren:

```
import music

tune = ["C4:4", "D", "E", "C", "C", "D", "E", "C", "E", "F",
        "G:8", "E:4", "F", "G:8"]
music.play(tune)
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/music.html#wolfgang-amadeus-microbit>

“C4:4” bezeichnet die Note “C” in Oktave Nummer 4 (Oktaven reichen von 0 bis 8. C4 ist also das mittlere C), gespielt für eine Dauer von 4. Pausen werden mit “R” bezeichnet (= rest). Wie du im Beispiel siehst, muss man Oktaven oder Längen nur bei Wechseln angehen. Das spart viel Schreibarbeit.

## Klangeffekte

```
import music

while True:
    for freq in range(880, 1760, 16):
        music.pitch(freq, 6)
```

```
for freq in range(1760, 880, -16):
    music.pitch(freq, 6)
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/music.html#sound-effects>

Dieser Polizeisireneneffekt zeigt, dass man auch Töne erzeugen kann, die keine Musiknoten sind - nervig! Die Methode *music.pitch()* erhält als erstes Argument die Frequenz in Hertz (Hz). 440 bezeichnet den Kammerton "a".



Die Funktion *range()* im Beispiel erwartet drei Argumente: Startwert, Endwert und Schrittweite. *range(1760, 880, -16)* bedeutet also "Erzeuge eine Liste mit Zahlen von 1760 bis 880 in Schritten zu 16", also 1760, 1744, ..., 896, 880.

Hier haben wir zwei Schleifen verwandt. Zwei innere *for*-Schleifen erzeugen den auf- und abschwellenden Ton. Die umschließende *while*-Schleife sorgt dafür, dass die Sirene nicht aufhört.

## Zufall

```
from microbit import *
import random

names = ["Mary", "Yolanda", "Damien", "Alia", "Kushal",
         "Mei Xiu", "Zoltan" ]

display.scroll(random.choice(names))
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/random.html#random>

Die Methode *random.choice()* wählt zufällig einen aus der Liste mit sieben Namen aus. Das Ergebnis wird an *display.scroll()* übergeben. Die Übergabe funktioniert wie Zwiebelschalen von innen nach außen.

## Zufallszahlen

```
from microbit import *
import random

display.show(str(random.randint(1, 6)))
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/random.html#random-numbers>

Das Beispiel realisiert einen einfachen Würfel. Nach jedem Reset wird eine neue Zahl angezeigt.

Es gibt diese Methoden im Modul *random* zum Erzeugen von Zufallszahlen:

- *random.randint()* - ganzzahlige Zufallszahl zwischen den beiden Argumenten (einschließlich)
- *random.randrange()* - ganzzahlige Zufallszahl zwischen 0 und dem einzelnen Argument (ohne den Argumentwert selbst)
- *random.random()* - Zufallszahl zwischen und 0.0 und 1.0 (einschließlich)

## Vorherbestimmte Zufallszahlen

```
from microbit import *
import random

random.seed(1337)
while True:
    if button_a.was_pressed():
        display.show(str(random.randint(1, 6)))
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/random.html#seeds-of-chaos>

Dieses Beispiel realisiert einen Würfel, der immer dieselbe Zahlenfolge erzeugt.

 Warum reicht es in diesem Beispiel nicht, wie im vorigen Beispiel beim Reset eine neue Zahl zu erzeugen?

# Stunde #9 - Noch mehr Funktionen

## Weitere Beispiele

### Bewegung

```
from microbit import *

while True:
    reading = accelerometer.get_x()
    if reading > 20:
        display.show("R")
    elif reading < -20:
        display.show("L")
    else:
        display.show("-")
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/movement.html#movement>

Bewegungen werden mit dem eingebauten Accelerometer gemessen:

- X - links / rechts
- Y - vorwärts / rückwärts
- Z - hoch und runter

Smartphones verwenden ein eingebautes Accelerometer, um den Bildschirm je nach Haltung zu drehen. Auch Game Controller enthalten Accelerometer.

Mit Micropython lassen sich Methoden sehr einfach kombinieren. Hier ist ein einfaches Musikinstrument:

```
from microbit import *
import music

while True:
    music.pitch(accelerometer.get_y(), 10)
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/movement.html#musical-mayhem>



Wie könnte man dieses Instrument musikalischer klingen lassen?

## Gesten

```
from microbit import *

while True:
    gesture = accelerometer.current_gesture()
    if gesture == "face up":
        display.show(Image.HAPPY)
    else:
        display.show(Image.ANGRY)
```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/gestures.html#gestures>

MicroPython kennt die folgenden Gesten: up, down, left, right, face up, face down, freefall, 3g, 6g, 8g, shake.

## Magic-8-Ball<sup>48</sup>

```
from microbit import *
import random

answers = [
    "Es ist sicher",
    "Es ist definitiv so",
    "Ohne Zweifel",
    "Ja, unbedingt",
    "Du kannst dich darauf verlassen",
    "Meiner Meinung nach ja",
    "Sehr wahrscheinlich",
    "Sieht gut aus",
    "Ja",
    "Die Zeichen stehen auf ja",
    "Ich bin nicht sicher. Noch einmal",
    "Frage später noch einmal",
    "Das will ich nicht verraten",
    "Kann ich gerade nicht sagen",
    "Konzentriere dich. Noch einmal",
    "Verlass dich nicht drauf",
    "Meine Antwort ist nein",
    "Meine Quellen sagten nein",
    "Die Aussichten sind gut",
    "Sehr zweifelhaft",
]

while True:
    display.show("8")
    if accelerometer.was_gesture("shake"):
        display.clear()
        sleep(1000)
        display.scroll(random.choice(answers))
```



<https://microbit-micropython.readthedocs.io/en/latest/tutorials/gestures.html#magic-8>



Wie kannst du einen Cheat einbauen (Tipp: Benutze die Taster)?

## Richtung - Der Kompass

```
from microbit import *
```

<sup>48</sup> [https://de.wikipedia.org/wiki/Magic\\_8\\_Ball](https://de.wikipedia.org/wiki/Magic_8_Ball)

```

compass.calibrate()

while True:
    needle = ((15 - compass.heading()) // 30) % 12
    display.show(Image.ALL_CLOCKS[needle])

```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/direction.html#compass>

Vor der Nutzung des Kompass muss man in kalibrieren. Das kleine Programm nimmt das Ergebnis von `compass.heading()` (Eine ganze Zahl zwischen 0 und 360) und berechnet daraus mit den Funktionen Modulo und ganzzahliger Division einen Uhrenzeiger.

Beispielrechnungen:

- Nord. `compass.heading() == 0. (15 - 0) // 30 == 0. 0 % 12 == 0.` Zeiger auf 12 Uhr
- Ost. `compass.heading() == 90. (15 - 90) // 30 == -3. -3 % 12 == 9.` Zeiger auf 9 Uhr
- Süd. `compass.heading() == 180. (15 - 180) // 30 == -6. -6 % 12 = 6.` Zeiger auf 6 Uhr
- Nord-West. `compass.heading() == 315. (15 - 315) // 30 == -10. -10 % 12 = 2.` Zeiger auf 2 Uhr

## Dateisystem

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/storage.html#storage>

... nicht behandelt.

## Sprachausgabe

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/speech.html#speech>

... nicht behandelt

## Netzwerk

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/network.html#network>

... nicht behandelt

## Radio

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/radio.html#radio>

... behandeln wir später

## Stunde #10 - Größere Programmbeispiele

Der Quellcode von MicroPython enthält einen ganzen Ordner mit Beispielprogrammen:<sup>49</sup>

analog_watch.py	Eine Uhr mit Zeigern und blinkendem Punkt
-----------------	---

---

<sup>49</sup> <https://github.com/bbcmicrobit/micropython/tree/master/examples>

asmleds.py	Technische Demonstration der LED-Anzeige
bubble_level_2d.py	Zeigt die Lage der Platine mit einem Leuchtpunkt
compass.py	Zeigt nach Norden
conway.py	Das Spiel des Lebens von John Horton Conway <sup>50</sup>
counter.py	Einfacher Zähler
digital_water.py	Eine Simulation von Wasser auf dem Display
dodge_game.py	Ein Ausweichspiel
flame_simulation.py	Anzeige von Flammen auf dem Display
flappybit.py	Das Spiel
four_buttons.py	Simuliert zwei weitere Taster über Analog-Eingänge
i_feel_today.py	Zeigt eine Skala von Emotionen auf dem Display
led_dance.py	Tanzende LEDs mit Ausblend-Effekt
magic8.py	Zufällige Antworten auf ja/nein-Fragen
maze.py	Per Accelerometer durch ein Labyrinth steuern
music.py	Spiele eine Melodie
neopixel_random.py	Zufallsfarben auf angeschlossenen NeoPixel-LEDs
play_file.py, reverb.py	Spiele Audio-Dateien
pomodoro.py	Ein Pomodoro-Timer zum konzentrierten Arbeiten
radio.py	Glühwürmchen
simple_slalom.py	Ein Spiel mit dem Accelerometer
speech.py	“Singen” mit einer Computerstimme
tiltmusic.py	Ein Musikinstrument mit dem Accelerometer
watch.py	Eine Uhr. Anzeige von Minuten & Stunden. Uhrzeit setzen
waveforms.py	Spiele verschiedene Wellenformen als Audio

<sup>50</sup> [https://de.wikipedia.org/wiki/Conways\\_Spiel\\_des\\_Lebens](https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens)

## Stunde #11 - Der heiße Draht in Python

### Erste Version - Berührungen zählen

```
# heisser-draht-01.py
from microbit import *

while True:
    if button_a.was_pressed():
        display.show(Image(
            "99909:",
            "90909:",
            "90909:",
            "90999:",
            "90000"
        ))
        sleep(1000)

    beruehrt = 0
    display.show(str(beruehrt))

    if pin0.is_touched():
        display.show(Image(
            "90009:",
            "09090:",
            "00900:",
            "09090:",
            "90009"
        ))
        sleep(1000)

    beruehrt += 1

    display.show(str(beruehrt))
```

<https://github.com/tinkerthon/Der-heisse-Draht-2017/blob/master/heisser-draht-01.py>

### Zweite Version - Berührungen & Zeit zählen

```
# heisser-draht-02.py
from microbit import *
import music

beruehrt = 0
```

```

jetzt = 0

while True:
    if button_a.was_pressed():
        music.play(music.POWER_UP)
        display.show(Image(
            "99909:"
            "90909:"
            "90909:"
            "90999:"
            "90000"
        ))
        sleep(1000)

    beruehrt = 0
    jetzt = running_time() // 1000
    display.show(str(beruehrt))

    if pin1.is_touched():
        display.show(Image(
            "90009:"
            "09090:"
            "00900:"
            "09090:"
            "90009"
        ))
        sleep(1000)

    beruehrt += 1

    display.show(str(beruehrt))

    neues_jetzt = running_time() // 1000
    if jetzt > 0 and neues_jetzt > jetzt:
        jetzt = neues_jetzt
        music.pitch(440, 6)

```

<https://github.com/tinkerthon/Der-heisse-Draht-2017/blob/master/heisser-draht-02.py>



Wie kannst du Berührungen und Zeit gleichzeitig anzeigen? Tipp: Beispiel  
“watch.py”

## Stunde #12 - Weitere Versionen

### Dritte Version - Ergebnis anzeigen

```
# heisser-draht-03.py
from microbit import *
import music

beruehrt = 0
jetzt = 0

while True:
    if button_a.was_pressed():
        music.play(music.POWER_UP)
        display.show(Image(
            "99909:",
            "90909:",
            "90909:",
            "90999:",
            "90000"
        ))
        sleep(1000)

        beruehrt = 0
        start = jetzt = running_time() // 1000
        display.show(str(beruehrt))

    if button_b.was_pressed():
        music.play(music.POWER_DOWN)
        display.scroll(
            str(beruehrt) + ':' +
            str(jetzt - start) + '=' +
            str(jetzt - start + beruehrt * 5)
        )
        jetzt = 0

    if pin1.is_touched():
        display.show(Image(
            "90009:",
            "09090:",
            "00900:",
            "09090:",
            "90009"
        ))
        sleep(1000)
```

```

beruehrt += 1

display.show(str(beruehrt))

neues_jetzt = running_time() // 1000
if jetzt > 0 and neues_jetzt > jetzt:
    jetzt = neues_jetzt
    music.pitch(440, 6)

```

<https://github.com/tinkerthon/Der-heisse-Draht-2017/blob/master/heisser-draht-03.py>

## Stunde #13 - Funkverbindung

Der Micro:bit hat Hardware an Bord, mit der sich Bluetooth-Verbindungen oder einfachere Radio-Verbindungen aufbauen lassen.

Bluetooth wird von Micropython nicht unterstützt, deshalb schauen wir uns das nicht näher an. Für die Programmierung kannst du aber PXT benutzen, womit wir am ersten Tag gestartet sind. Wenn ihr zuhause damit experimentieren wollt: Dieser Android-Entwickler<sup>51</sup> hat etwas Material zusammengetragen. Sei aber gewarnt. Die Bluetooth-Verbindungen scheinen nicht ganz unproblematisch zu sein.<sup>52</sup>

### Micro:bit als Glühwürmchen

Zur Einstimmung auf die Funkverbindungen installiert bitte alle einmal das folgende Programm von Nicholas H. Tollervey. Der hat auch den Mu-Editor programmiert.

```

# Ein micro:bit Glühwürmchen.
# Von Nicholas H.Tollervey. In der Public Domain.

import radio
import random
from microbit import display, Image, button_a,
sleep

# Erzeuge die "flash" Animation Frames
flash = [
    Image().invert()*(i/9)
    for i in range(9, -1, -1)
]

```

---

<sup>51</sup> <http://bluetooth-mdw.blogspot.de/p/bbc-microbit.html>

<sup>52</sup>

<https://www.golem.de/news/bbc-microbit-im-test-schulrechner-muss-noch-dazulernen-1607-121757-3.html>

```

# Das Radio muss eingeschaltet werden.
radio.on()

# Schleife mit Ereignissen ("Event Loop").
while True:
    # Taster A sende eine "flash"-Nachricht.
    if button_a.was_pressed():
        radio.send('flash')  # a-ha

    # Read any incoming messages.
    incoming = radio.receive()
    if incoming == 'flash':

        # Bei eingehender "flash"-Nachricht
        # startet die Animation nach einer
        # zufälligen, kurzen Pause
        sleep(random.randint(50, 350))
        display.show(flash, delay=100, wait=False)

        # Zufällig wird eine weitere Nachricht
        # nach kurzer Wartezeit versandt.
        if random.randint(0, 9) == 0:
            sleep(500)
            radio.send('flash')  # a-ha

```

<https://microbit-micropython.readthedocs.io/en/latest/tutorials/radio.html>

Mit 15 Micro:bits sollte das ein tolles Schauspiel geben.

#### **Vierte Version: Ergebnisse an Zentrale für High Score melden**

Zunächst einmal: Wir werden hier nicht versuchen, ein komplettes Programm zu entwickeln.

Wenn du hier Ambitionen entwickelst, schaue dir Kivy<sup>53</sup> an. Damit kannst du mit Python Windows-Programme ebenso entwickeln wie Apps für Android oder iOS.

Für eine zentrale Sammlung erweitern wir unser Programm um wenige Zeilen für das Radio-Modul:

- Füge oberhalb der While-Schleife

```

import radio
radio.on()
... ein.

```

- In der If-Anweisung für Taster B, sammle das Ergebnis zunächst in einer Variable und gebe diese dann sowohl per Funk als auch über das Display aus:

```

radio.send(ergebnis)
display.scroll(ergebnis)

```

---

<sup>53</sup> <https://kivy.org/>

Auf der Empfängerseite können wir ein komplettes Programm schreiben. Der Lehrer Jez Dean<sup>54</sup> hat dazu eine Anleitung geschrieben. Wir machen es uns aber einfach und benutzen folgendes kurze Skript zusammen mit dem Programm pyboard aus dem Micropython Quellcode.<sup>55</sup>

```
# Einfache Protokollierung von Funksendungen.

import radio
from microbit import *

radio.on()

while True:
    incoming = radio.receive()
    if (incoming != None):
        print(str(running_time()) + ': ' +
incoming)
```

Dieses Programm braucht auch nur einmal auf dem Rechner des Workshop-Leiters laufen.

### Fünfte Version: Eine zentrale Rangliste

Ich hatte es schon erwähnt: Python läuft nicht nur auf dem Micro:bit, sondern ist auch hervorragend für alle möglichen Anwendungen geeignet, z.B. zum Erstellen einer Datenbankgestützten Website. Die vierte Version unseres Programms sandte bereits die Spielergebnisse an einen zentralen Micro:bit. Wäre es nicht toll, diese Ergebnisse von dort aus als Rangliste auf einer Website darzustellen?

## Der heiße Draht - Rangliste

Stand: 2017-04-13 um 22:50:16 Uhr

Position	Spieler	Anzahl	Ø Zeit	Ø Berührungen	Ø Punkte
1	Olav	6	10	1	12.67
2	#12	2	15	2	24.5
3	Tralla	3	9	5	32.67

<sup>54</sup>

Rangliste | Namen | Github

Damit können wir dann ein Turnier veranstalten!

Unser Programm auf dem Micro:bit braucht noch einmal ein paar Erweiterungen. Hier ist das vollständige Programm:

```
# heisser-draht-05.py

from microbit import *
import music
import radio

beruehrt = 0
jetzt = 0
start = 0

radio.on()

display.show(Image(
    "90990:"
    "90909:"
    "90909:"
    "90909:"
    "90990"
))

sleep(1000)
id = 1

while not button_b.was_pressed():
    display.show(str(id))
    if button_a.was_pressed():
        id += 1

    if id > 15:
        id = 1

music.play(music.BA_DING)

while True:
    if button_a.was_pressed():
        music.play(music.POWER_UP)
        display.show(Image(
            "99909:"
            "90909:"
            "90909:"
            "90999:"
            "90000"
        ))

```

```

sleep(1000)
beruehrt = 0
button_b.was_pressed()
start = jetzt = running_time() // 1000
display.show(str(beruehrt))

if jetzt > 0 and button_b.was_pressed():
    ergebnis = str(id) + ':' +
        str(beruehrt) + ':' +
        str(jetzt - start) + ':' +
        str(jetzt - start + beruehrt * 5)

    radio.send(ergebnis)

    music.play(music.POWER_DOWN)
    display.scroll(ergebnis)
    jetzt = 0

    sleep(1000)
    display.show(str(id))

if pin1.is_touched():
    display.show(Image(
        "90009:"
        "09090:"
        "00900:"
        "09090:"
        "90009"
    ))
    sleep(1000)
    beruehrt += 1
    display.show(str(beruehrt))

neues_jetzt = running_time() // 1000
if jetzt > 0 and neues_jetzt > jetzt:
    jetzt = neues_jetzt
    music.pitch(440, 6)

```

Für das Sammeln der Ergebnisse brauchen wir folgende Software:

- Python 3<sup>56</sup>
- VirtualEnv & Pip<sup>57</sup>

---

<sup>56</sup> <https://www.python.org/downloads/>

<sup>57</sup> <https://virtualenv.pypa.io/en/stable/>

- Das Web-Framework Bottle<sup>58</sup> mit SQLite-Plugin<sup>59 60</sup>
- Das Python-Modul pySerial<sup>61</sup>
- Die CSS-Framework Milligram<sup>62</sup>

Installation:

```
virtualenv -p python3 env
source env/bin/activate.fish
pip install --upgrade pip
pip install bottle bottle-sqlite
pip install pySerial

# Im ersten Fenster sammelt score.py die Ergebnisse
python score.py

# Im zweiten Fenster läuft die Website
cd leaderboard
python index.py
```

Einfacher für den Workshop-Leiter, aber noch etwas komplizierter im Aufsetzen ist, das Python-Programm in einen Webserver per uWSGI zu integrieren.<sup>63</sup>

Obwohl ich möglichst einfache Bausteine benutzt habe, geht das Programm zum Sammeln und Anzeigen der Rangliste doch über das hinaus, was wir uns im Rahmen des Workshops anschauen können. Die Quellen der Programme liegen aber als OpenSource in meinem Github-Konto.<sup>64</sup> Hier grob die Funktionsweise:

- Unsere Micro:bits senden mit dem Abschluss eines jeden Spiels eine Zeichenkette über die Methode `radio.send()`. Darin enthalten sind die ID des Spielers, die Zahl der Berührungen, die Zeit und eine Punktezahl, berechnet aus der Zeit und je 5 Strafsekunden pro Berührung. Die Rangliste wird aus diesen Punkten gebildet. Je weniger Punkte, desto besser.
- Ein besonderer Micro:bit ist mit dem Rechner des Workshop-Leiters verbunden. Auf dem Micro:bit läuft das Protokollierungsprogramm, welches wir schon bei Variante #4 benutzt haben.
- Auf dem Rechner laufen zwei weitere Programme, die über eine kleine Datenbank miteinander verbunden sind. Diese Datenbank speichert für jeden Spieler die eingehenden Werte und die Anzahl gespielter Spiele.

---

<sup>58</sup> <https://bottlepy.org/>

<sup>59</sup> <http://bottlepy.org/docs/0.9/plugins/sqlite.html>

<sup>60</sup> <https://docs.python.org/3.6/library/sqlite3.html>

<sup>61</sup> <https://pypi.python.org/pypi/pyserial>

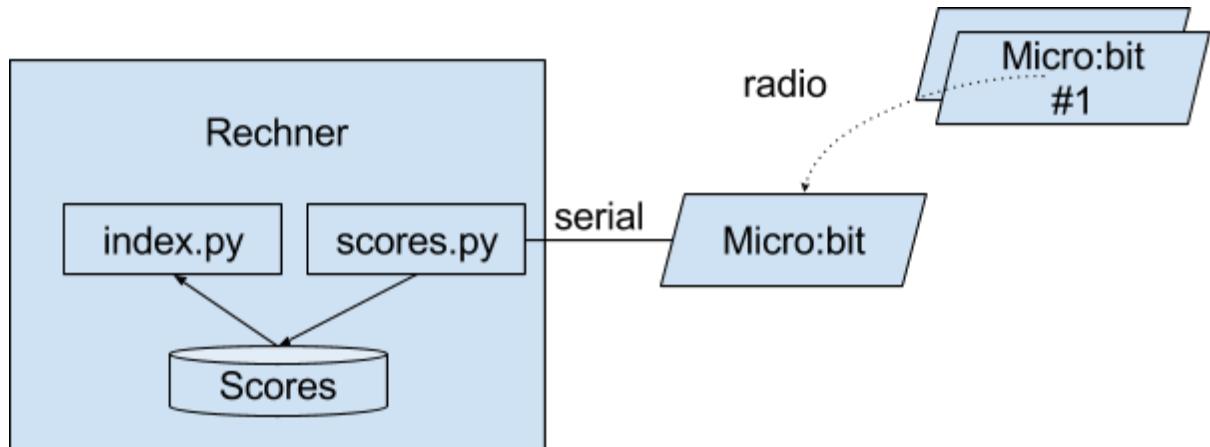
<sup>62</sup> <http://milligram.github.io/>

<sup>63</sup> <https://askubuntu.com/questions/590027/how-to-set-python-3-as-default-interpreter-in-ubuntu-14-4>

<sup>64</sup> <https://github.com/tinkerthon/Der-heisse-Draht-2017>

- Das erste Programm steht mit dem angeschlossenen Micro:bit in Verbindung und speichert die eingehenden Spielergebnisse in der Datenbank.
- Das zweite Programm erstellt aus dieser Datenbank eine Webseite, auf der die Rangliste dargestellt wird.
- Zusätzlich lässt sich über diese Webseite noch die Liste der Teilnehmer pflegen, um die eingehenden IDs als Namen anzeigen zu können.

Hier ist der Aufbau im Überblick:



## The End

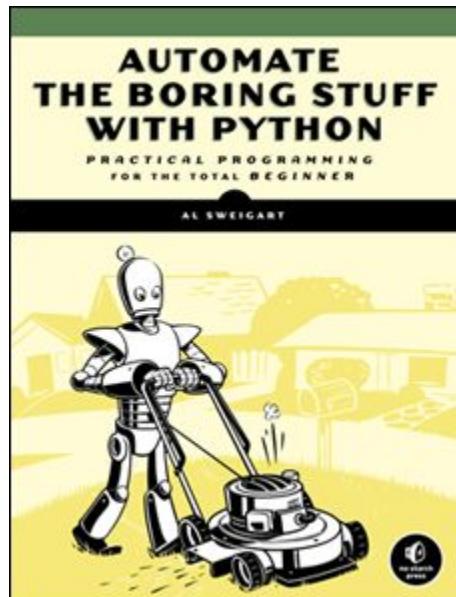
Jetzt hast du einmal in die Welt von Python und Software-Entwicklung reingeschnuppert. Es gibt aber noch viel zu entdecken. Wie gesagt: Mit Python kann man eine Menge anstellen:

- Apps für Windows, Mac, Android oder iPhone entwickeln mit **Kivy**<sup>65</sup>
- Websites entwickeln mit **Django**<sup>66</sup>

<sup>65</sup> <https://kivy.org/#home>

<sup>66</sup> <https://www.djangoproject.com/>

- **Spiele** entwickeln oder Langweiliges automatisieren. Bei InventWithPython<sup>67</sup> gibt es kostenlose, englische Bücher zum Thema.



- Sprachverarbeitung und künstliche Intelligenz. Kennst du **Amazon Alexa**? Dafür kannst du in Python<sup>68</sup> oder Javascript eigene Skills schreiben. Im Buch "Kollektive Intelligenz"<sup>69</sup> sind die meisten Beispiele in Python geschrieben
- Zu Python allgemein gibt es natürlich auch jede Menge Bücher. Schaut einfach mal in einem Buchladen oder der Stadtbücherei nach.
- Eine gute Möglichkeit, Pythonistas kennenzulernen, ist immer die **Froscon**-Konferenz mit den FrogLabs in St. Augustin im Sommer.<sup>70</sup>
- Interessierst du dich mehr für die **Hardware**? Bei Thalia in Bonn ist am Samstag, dem 3. Juni 2017 eine Mini MakerFaire. In diesem Fall ist auch das Büchlein von Burkhard Kainka "Micro:bit Praktikum"<sup>71</sup> interessant für dich. Er benutzt den Micro:bit wie ein kleines Elektronik-Labor. Sogar ein einfaches Oszilloskop wird gebaut.

<sup>67</sup> <http://inventwithpython.com/>

<sup>68</sup>

<https://developer.amazon.com/blogs/post/Tx14R0IYYGH3SKT/Flask-Ask-A-New-Python-Framework-or-Rapid-Alexa-Skills-Kit-Development>

<sup>69</sup> <http://amzn.to/2oyf0qO>

<sup>70</sup> <https://www.froscon.de/startseite/>

<sup>71</sup> <http://amzn.to/2oKcJZv>



- Wenn du Zubehör zu deinem Micro:bit suchst: Bei Amazon gibt es inzwischen einiges.<sup>72</sup>

## Materialien

- Acht Windows-Laptops (einer pro zwei Teilnehmer) mit installierter Programmierumgebung für den Editor Mu<sup>73</sup>
- WLAN-Zugang für den Workshop-Leiter und die Windows-Laptops

			<b>Einzelpreis inkl. MwSt. zzgl. Versand</b>		<b>Bestellnummer</b>	<b>Bauteil</b>
Pro Teilnehmer	€17.70	0.1	€177.00	810 601		BBC Micro:bit Club MB224-Set, 10 Stück mit Zubehör <sup>74</sup>
	€1.01	0.3	€3.35	830 595		Messleitungen mit Krokodilklemmen, 0,5 mm <sup>2</sup> , 300 mm, 10 Stück <sup>75</sup>

<sup>72</sup> <http://amzn.to/2pfY5Mn>

<sup>73</sup> <https://codewith.mu/>

<sup>74</sup> [http://www.pollin.de/shop/dt/ODkzOTgxOTk-/Bauelemente\\_Bauteile/Entwicklerboards/Sonstige\\_Boards/BBC\\_Micro\\_bit\\_Club\\_MB224\\_Set\\_10\\_Stueck\\_mit\\_Zubehoer.html](http://www.pollin.de/shop/dt/ODkzOTgxOTk-/Bauelemente_Bauteile/Entwicklerboards/Sonstige_Boards/BBC_Micro_bit_Club_MB224_Set_10_Stueck_mit_Zubehoer.html)

<sup>75</sup> [http://www.pollin.de/shop/dt/NDA0OTYxOTk-/Messtechnik/Messzubehoer/Leitungen\\_Spitzen\\_Klemmen/Messleitungen\\_mit\\_Krokodilklemmen\\_0\\_5\\_mm\\_300\\_mm\\_10\\_Stueck.html](http://www.pollin.de/shop/dt/NDA0OTYxOTk-/Messtechnik/Messzubehoer/Leitungen_Spitzen_Klemmen/Messleitungen_mit_Krokodilklemmen_0_5_mm_300_mm_10_Stueck.html)

	€0.75	1	€0.75	390 116	Piezo-Schallwandler SOUNDTECH SEP-1110
Preis pro Teilnehmer	€19.46				
Anzahl Teilnehmer		20			
Preis bei Teilnehmerzahl	€389.10				
Zusätzlich pro Teilnehmer (Baumarkt)					Gehobelte Dachlatte, 40 x 15 mm Querschnitt, 20 cm lang für den Stand mit 2 Bohrungen für Spieldraht  Rundstäbe, 10 mm Durchmesser und 8 cm lang für den Griff mit 1 Bohrung für Öse  ca. 50cm Verzinkter Eisendraht <sup>76</sup>

## Historie des Workshops

### Bisher:

- 8 Kinder, 9-12 Jahre
- 2 Tage
- thematisch: Spiel "Der heiße Draht"
- Ablauf: Platine löten & Spielaufbau, Programmierung der Spielsteuerung
- technische Basis: PICAXE Microcontroller, separates Programmierkabel, Laptops mit je zwei Kindern
- Materialkosten ~10€ für Bauteile + Platine, ~20€ für Programmierkabel

### Neu:

- 15 Kinder, ab 8. Klasse (13 - 16 Jahre)
- 3 Tage
- thematisch: Spiel "Der heiße Draht"
- Ablauf: Spielaufbau, Funktionen der Platine, Kennenlernen der Programmierumgebungen, Einführung in Programmierung, Programmierung der Spielsteuerung
- technische Basis: BBC micro:bit, Laptops für je 2-3 Kinder, Handy / Tablet
- Materialkosten: ~20€ (nur Platine) - 30€ (Platine, Batteriebox, USB-Kabel)

<sup>76</sup><https://www.knauber-freizeit.de/Heimwerken/Eisenwaren/Befestigen-%26-Verbinden/Eisendraht-verz-1%2C60mm/p/3110101245>

# Über den BBC micro:bit

Die BBC micro:bit Microcomputer-Platine wurde in der ersten Jahreshälfte 2016 von der BBC an 1 Million Britischer Schüler im Alter von 11-12 Jahren kostenlos verteilt (Artikel von Sinead Rocks. Leiterin von BBC Learning, vom 30.5.2016<sup>77</sup>).

Der BBC micro:bit ist Teil einer groß angelegten Initiative der BBC, die damit, unterstützt von mehr als 20 Partnern, eine erwartet Lücke von 1,4 Millionen Digitalarbeitern in Großbritannien in den nächsten 5 Jahren schließen will.

Der BBC micro:bit schließt eine wichtige Lücke in der Landschaft der Mikrocontroller-Boards, wie sie für die Ausbildung eingesetzt werden. Boards wie der Arduino oder auch der PICAXE, die wir in den vergangenen Jahren in den Physical Computing Workshops eingesetzt haben, erfordern weitere Bauteile oder sogar Lötarbeiten, um erste Experimente zum Messen und Steuern der Umwelt durchzuführen. Am anderen Ende der Leistungsfähigkeit hat sich der Raspberry Pi als kompletter Desktop-Rechner etabliert, der zwar viele Ein- und Ausgänge besitzt, aber ebenfalls Zusatzschaltungen für Mikrocontroller-Experimente benötigt.

Der BBC micro:bit hat ein, wenn auch kleines, so doch vielfältiges Arsenal an Ein- und Ausgängen direkt auf der Platine, so dass mit ihm ohne weitere Bauteile kleine Spiele entwickelt werden können:

- 25 rote LEDs für Nachrichten, Symbole, Spiele oder digitale Geschichten
- zwei programmierbare Taster. Damit wird der micro:bit zum Game Controller oder kann eine Musik-Playlists steuern
- ein Bewegungsfühler („Accelerometer“), mit dem sich Schütteln, Kippen oder Beschleunigung in Programmen erkennen lassen
- ein eingebauter Kompass („Magnetometer“). Damit lassen sich Richtungsänderungen und bestimmte Metallarten erkennen
- Bluetooth zur Vernetzung von micro:bits untereinander oder mit Handys oder Tablets. Damit lassen sich Handy-Funktionen steuern oder komplexe Netzwerke aufbauen
- Fünf Ein-/Ausgaberinge für Krokodilklemmen zum einfachen Anschluß an externe Geräte und Messfühler zur Steuerung

Zusätzliche Bauteile sind einfach über Krokodilklemmen abschließbar. Das werden wir für unser Spiel „Der heiße Draht“ nutzen.

Darüber hinaus hat der micro:bit Elektronik für eine Vernetzung über Bluetooth direkt eingebaut. Damit ist die Steuerung über Handy oder die drahtlose Verbindung mehrerer micro:bit einfach möglich.

Die Programmierung ist über ein Standard-USB-Kabel oder sogar per Bluetooth vom Handy/Tablet aus möglich.

Hier ein kurzes Video zu der Platine: <https://www.youtube.com/watch?v=Wuza5WXiMkc>

---

<sup>77</sup> <http://www.bbc.co.uk/blogs/aboutthebbc/entries/8170eb55-edbd-4fa7-9b8d-855740cdf763>

Hier noch weiterführende Links zum BBC micro:bit

- <https://www.microbit.co.uk/>
- <http://www.bbc.co.uk/makeitdigital>
- <http://www.bbc.co.uk/programmes/articles/1gkwk58DPmRzt2TzDp3pr9x/about-make-it-digital>
- <http://www.heise.de/make/meldung/Der-Billig-Computer-BBC-Micro-Bit-kommt-in-den-freien-Verkauf-3225084.html>

## Alternative Plattformen

- OCTOPUS - <https://www.tindie.com/products/FabLab/octopus-the-iot-badge/>
- Bayduino - <http://bayduino.com/>
- B-O-B-3 - <http://www.bob3.org/>
- Calliope Mini - <http://calliope.cc/>

## Der heiße Draht - Physical Computing<sup>78</sup>



für Schülerinnen und Schüler ab der 8. Klasse mit  
Olav Schettler am **19. bis 21. April 2017**

Kleinste Computer, so genannte Microcontroller, finden sich heute in fast allen elektronischen Geräten wie z.B. im Kühlschrank, Mikrowelle, elektrischer Zahnbürste, MP3-Spieler, Handy oder sprechendem Kinderspielzeug. Wir nutzen diese Geräte meist selbstverständlich und machen uns keine Gedanken über die Computertechnik in Alltagsgegenständen.

Im Workshop »Physical Computing« bauen die Kinder je ein eigenes Spiel »Der heiße Draht«.

Sie nutzen dazu den BBC micro:bit, einen leistungsfähigen Microcomputer. Die Steuerung des Spiels wird in der einfach zu erlernenden Programmiersprache Python geschrieben.

Die gebauten Spiele dürfen mit nach Hause genommen werden und sind ohne Computer lauffähig. Die Programmierumgebung ist kostenlos im Internet erhältlich und so kann das Spiel auch zuhause verändert oder weiterentwickelt werden. Dafür ist lediglich ein normales USB-Kabel nötig.

Höchstteilnehmerzahl: 15 Teilnehmer

Dauer: 3 x 6 Stunden (von 10 bis 16 Uhr)

<sup>78</sup><http://www.deutsches-museum.de/bonn/information/fuer-kinder-und-schulen/die-kleine-eule-pfififikus/workshops/der-heisse-draht/>

Wir freuen uns, dass dieser Workshop aus Landesmitteln im Rahmen von zdi NRW (Zukunft durch Innovation) gefördert wird. Daher ist der Workshop für Schülerinnen und Schüler ab der 8. Klasse, die in Nordrhein-Westfalen zur Schule gehen, **kostenlos**.

Zusätzlich zu einer verbindlichen schriftlichen Anmeldung an das Museum(hier) muss dafür außerdem von den Eltern das kurze zdi-Formular ausgefüllt und unterschrieben werden. Dieses senden wir Ihnen gerne mit der Buchungsbestätigung zu. Bitte mailen Sie uns dieses vor Beginn des Workshops oder geben Sie es spätestens am ersten Workshoptag Ihrem Kind mit.

Die Schülerinnen und Schüler können nur nach schriftlicher Voranmeldung durch die Eltern teilnehmen.

Eine Teilnahme am Workshop ohne die oben genannten Bedingungen ist ebenfalls möglich. In diesem Fall betragen die Kosten 150 Euro.

Die Teilnehmer erhalten ein Mittagessen im Wissenschaftszentrum.